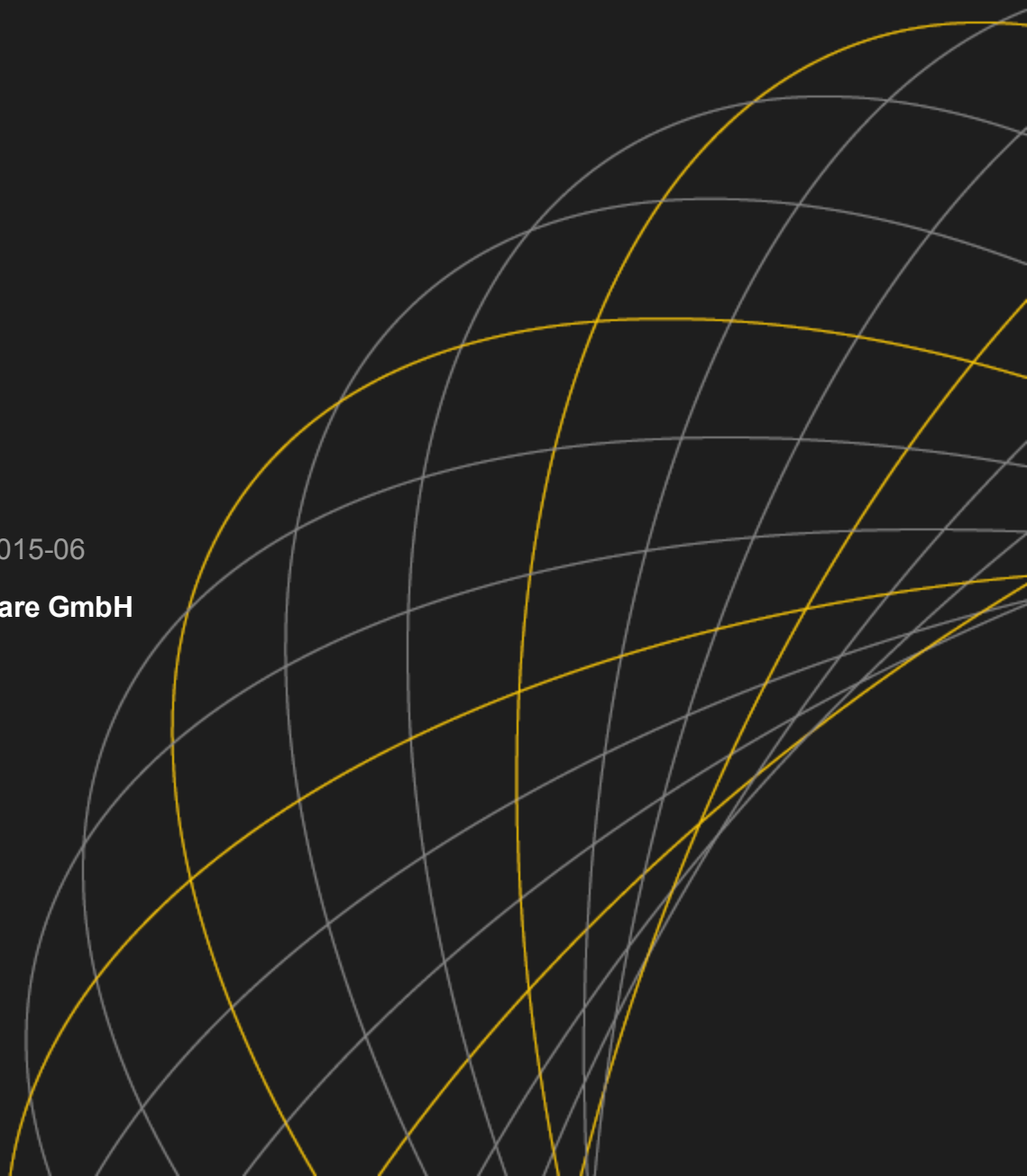
Automic™
Let's Automate Business.

Automation Engine 10

ONE Automation Platform

Automation Engine Script



Version: 10.0.5

Veröffentlicht: 2015-06

Automic Software GmbH

Copyright

Die Marke Automic® und das Automic-Logo® sind Warenzeichen der Automic Software GmbH (Automic). Die Nutzung sämtlicher Warenzeichen bedarf der ausdrücklichen schriftlichen Genehmigung und unterliegt den Lizenzbedingungen. Die Software/das Computerprogramm sind geschütztes Eigentum der Firma Automic. Zugang und Nutzung vorbehaltlich der geprüften schriftlichen Lizenzbedingungen.

Die Software/das Computerprogramm ist weiters durch internationale Verträge sowie nationale und internationale Gesetze urheberrechtlich geschützt. Unbefugter Zugang bzw. Nutzung werden zivil- und strafrechtlich verfolgt. Unbefugtes Kopieren und andere Formen der auszugsweisen oder vollständigen Reproduktion, Dekompilierung, Nachbau, Änderung oder die Entwicklung von aus der Software abgeleiteten Produkten sind strengstens verboten. Die Nichtbeachtung wird strafrechtlich verfolgt.

Alle Angaben ohne Gewähr. Änderungen, Irrtümer, Satz- und Druckfehler vorbehalten. Nachdruck auch in Auszügen ist untersagt.

© Copyright Automic Software GmbH. Alle Rechte vorbehalten.

Inhaltsverzeichnis

1 Einführung **1**
1.1 Über das AE Script 1
1.2 Erste Schritte 2
1.2.1 Einführung 2
1.2.2 Vorab-Checkliste 2
1.2.3 Das erste Script 3
1.2.4 Script-Anweisungen 4
1.2.5 Script-Variablen 5
1.2.6 Script-Funktionen 6
1.2.7 Verwendung von JCL 7
1.2.8 Benutzerdefinierte Dialoge 8
1.2.9 IF-Bedingungen 9
1.2.10 Schleifen im Script 11
1.3 Grundlagen 13
1.3.1 Scriptaufbau 13
1.3.2 Kontrollstrukturen 14
1.3.3 Script-Variable 15
1.3.4 Script-Literal 18
1.3.5 Script-Registerkarten 19
1.3.6 Scripteditor 20
1.3.7 Starthilfe 23
1.4 Weiterführende Themen 24
1.4.1 Script-Bausteine benutzen 24
1.4.2 Fehlerbehandlung 24
1.4.3 Verwendung von Variable-Objekten 25
1.4.4 Objekt-Attribute verändern 26
1.4.5 Rückgabewerte von Funktionen 26
1.4.6 Rechnen 27
1.4.7 Zeichenketten 29
1.4.8 Datums-, Zeit- und Periodenformate 30
1.4.9 Benachrichtigungen verschicken 32
1.4.10 Scriptverarbeitung 33

1.5 Nicht länger unterstützte Script-Sprachmittel 33
2 Script-Sprachmittel - alphabetische Auflistung 35
3 Funktionelle Gliederung 46
3.1 Script-Sprachmittel - funktionelle Gliederung 46
3.1.1 Systemzustände und -bedienung 54
3.2 Objekte bearbeiten 58
3.2.1 :REGISTER_OUTPUTFILE 58
3.2.2 CREATE_OBJECT 59
3.2.3 EXPORT 63
3.2.4 IMPORT 65
3.2.5 MODIFY_OBJECT 67
3.2.6 MOVE_OBJECT 71
3.2.7 REMOVE_OBJECT 73
3.3 Objekte aktivieren 74
3.3.1 :BEGINREAD... :ENDREAD 74
3.3.2 :PRINT 76
3.3.3 :PUT_READ_BUFFER, :PUT_PROMPT_BUFFER 78
3.3.4 :READ 80
3.3.5 ACTIVATE_UC_OBJECT 85
3.3.6 AUTOFORECAST 90
3.3.7 CANCEL_UC_OBJECT 91
3.3.8 DEACTIVATE_UC_OBJECT 93
3.3.9 FORECAST_OBJECT 94
3.3.10 FORECAST_TASK 96
3.3.11 RERUN_UC_OBJECT 98
3.3.12 RESTART_UC_OBJECT 99
3.3.13 ROLLBACK_UC_OBJECT 102
3.3.14 SYS_STATE_ACTIVE 103
3.3.15 SYS_ACTIVE_COUNT 104
3.3.16 SYS_STATE_JOB_ACTIVE 107
3.3.17 SYS_STATE_JOBS_IN_GROUP 108
3.3.18 SYS_STATE_JP_ACTIVE 109
3.3.19 TOGGLE_OBJECT_STATUS 110
3.4 Objekte lesen oder verändern 111

3.4.1 :ADD_ATT ... 111
3.4.2 :ADD_COMMENT ... 112
3.4.3 :ATTACH_SYNC ... 113
3.4.4 :DELETE_VAR ... 115
3.4.5 :MODIFY_STATE ... 116
3.4.6 :PUT_ATT ... 118
3.4.7 :PUT_ATT_APPEND ... 121
3.4.8 :PUT_VAR ... 122
3.4.9 :PUT_VAR_COL ... 125
3.4.10 :REMOVE_ATT ... 126
3.4.11 :REPLACE_STRUCTURE ... 128
3.4.12 :SET_CALE ... 129
3.4.13 :SET_CONDITION ... 130
3.4.14 :XML_CLOSE ... 132
3.4.15 GET_ATT ... 133
3.4.16 GET_ATT_PLAIN ... 134
3.4.17 GET_ATT_SUBSTR ... 136
3.4.18 GET_CONDITION ... 137
3.4.19 GET_CONNECTION ... 139
3.4.20 GET_LOGIN ... 140
3.4.21 GET_OBJECT_TYPE ... 141
3.4.22 GET_OH_IDNR ... 142
3.4.23 GET_PUBLISHED_VALUE ... 143
3.4.24 GET_STATISTIC_DETAIL ... 145
3.4.25 GET_SYNC ... 148
3.4.26 GET_VAR ... 149
3.4.27 MODIFY_TASK ... 151
3.4.28 MODIFY_UC_OBJECT ... 176
3.4.29 SET_SYNC ... 181
3.4.30 XML_BEAUTIFY ... 182
3.4.31 XML_GET_ATTRIBUTE ... 184
3.4.32 XML_GET_CHILD_COUNT ... 185
3.4.33 XML_GET_FIRST_CHILD ... 187
3.4.34 XML_GET_LAST_CHILD ... 188

3.4.35 XML_GET_NEXTSIBLING 190
3.4.36 XML_GET_NODE_NAME 192
3.4.37 XML_GET_NODE_TEXT 193
3.4.38 XML_OPEN 195
3.4.39 XML_PRINTINTOFILE 197
3.4.40 XML_SELECT_NODE 198
3.5 Script-Aufbau und -Prozessierung 200
3.5.1 :CLEAR 200
3.5.2 :CONST, :CONSTANT 201
3.5.3 :DATA 202
3.5.4 :DEFINE 203
3.5.5 :EXT_REPORT_OFF 206
3.5.6 :EXT_REPORT_ON 207
3.5.7 :FILL 208
3.5.8 :GENERATE 209
3.5.9 :IF... :ELSE... :ENDIF 212
3.5.10 :INCLUDE 214
3.5.11 :INC_SCRIPT 216
3.5.12 :JCL_CONCAT_CHAR 217
3.5.13 :JCL_SUBSTITUTE 218
3.5.14 :PSET 219
3.5.15 :PUBLISH 221
3.5.16 :RESTART 223
3.5.17 :RSET 225
3.5.18 :SET 227
3.5.19 :SET_SCRIPT_VAR 228
3.5.20 :SWITCH ... :CASE ... :OTHER ... :ENDSWITCH 231
3.5.21 :WAIT 233
3.5.22 :WHILE... :ENDWHILE 234
3.5.23 FIND 236
3.5.24 GET_SCRIPT_VAR 237
3.5.25 LENGTH 239
3.6 Fehlerbehandlung und Meldungen 240
3.6.1 :EXIT 240

3.6.2 :ON_ERROR 241
3.6.3 :SEND_MSG 243
3.6.4 :SEND_SNMP_TRAP 244
3.6.5 :SET_LAST_ERR 246
3.6.6 :STOP 248
3.6.7 GET_MSG_TXT 250
3.6.8 GET_MSG_TYPE 251
3.6.9 SEND_MAIL 252
3.6.10 SYS_LAST_ERR_INS 254
3.6.11 SYS_LAST_ERR_NR 256
3.6.12 SYS_LAST_ERR_SYSTXT 257
3.7 Aktivierungsdaten 258
3.7.1 GET_PARENT_NAME 258
3.7.2 GET_PARENT_NR 259
3.7.3 GET_PARENT_TYPE 260
3.7.4 GET_UC_OBJECT_STATUS 261
3.7.5 GET_UC_OBJECT_NR 263
3.7.6 SYS_ACT_HOST 264
3.7.7 SYS_ACT_JP 265
3.7.8 SYS_ACT_ME_NAME 266
3.7.9 SYS_ACT_ME_NR 267
3.7.10 SYS_ACT_ME_TYPE 268
3.7.11 SYS_ACT_PARENT_NAME 268
3.7.12 SYS_ACT_PARENT_NR 270
3.7.13 SYS_ACT_PARENT_TYPE 271
3.7.14 SYS_ACT_PREV_NAME 273
3.7.15 SYS_ACT_PREV_NR 274
3.7.16 SYS_ACT_PTTYP 274
3.7.17 SYS_ACT_RESTART 275
3.7.18 SYS_ACT_RESTART_COUNT 276
3.7.19 SYS_ACT_RESTART_ME_NR 277
3.7.20 SYS_ACT_TOP_NAME 278
3.7.21 SYS_ACT_TOP_NR 279
3.7.22 SYS_ACT_USERID 279

3.7.23 SYS_LAST_RESTART_POINT 280
3.7.24 SYS_LAST_RESTART_TEXT 281
3.7.25 SYS_RESTART_POINT 282
3.8 Benutzerdaten 284
3.8.1 IS_GROUP_MEMBER 284
3.8.2 SYS_ACT_CLIENT, SYS_USER_CLIENT 285
3.8.3 SYS_ACT_CLIENT_TEXT 285
3.8.4 SYS_USER_ALIVE 286
3.8.5 SYS_USER_DEP 287
3.8.6 SYS_USER_LNAME 288
3.8.7 SYS_USER_NAME 289
3.9 Datensequenzen 290
3.9.1 :CLOSE_PROCESS 290
3.9.2 :PROCESS... :TERM_PROCESS... :ENDPROCESS 291
3.9.3 CREATE_PROCESS 293
3.9.4 GET_PROCESS_INFO 294
3.9.5 GET_PROCESS_LINE 296
3.9.6 LOAD_PROCESS 301
3.9.7 PREP_PROCESS 302
3.9.8 PREP_PROCESS_AGENTGROUP 307
3.9.9 PREP_PROCESS_COMMENTS 310
3.9.10 PREP_PROCESS_DOCU 312
3.9.11 PREP_PROCESS_FILE 314
3.9.12 PREP_PROCESS_FILENAME 318
3.9.13 PREP_PROCESS_PROMPTSET 322
3.9.14 PREP_PROCESS_REPORT 324
3.9.15 PREP_PROCESS_REPORTLIST 328
3.9.16 PREP_PROCESS_VAR 332
3.9.17 PUT_PROCESS_LINE 335
3.9.18 SAVE_PROCESS 337
3.9.19 WRITE_PROCESS 338
3.10 Ereignisbehandlung 341
3.10.1 GET_CONSOLE, GET_EVENT_INFO 341
3.10.2 GET_FILESYSTEM 351

3.10.3 GET_WIN_EVENT ... 356
Beispiele ... 357
3.11 Systemzustände und -bedienung ... 357
3.11.1 :DISCONNECT ... 357
3.11.2 :SET_UC_SETTING ... 359
3.11.3 :SHUTDOWN ... 361
3.11.4 :TERMINATE ... 362
3.11.5 CHANGE_LOGGING ... 363
3.11.6 GET_UC_SERVER_NAME ... 365
3.11.7 GET_UC_SETTING ... 365
3.11.8 GET_UC_SYSTEM_NAME ... 368
3.11.9 ILM ... 369
Syntax ... 372
3.11.10 MODIFY_SYSTEM ... 373
3.11.11 SYS_BUSY_01 ... 377
3.11.12 SYS_BUSY_10 ... 378
3.11.13 SYS_BUSY_60 ... 379
3.11.14 SYS_HOST_ALIVE ... 380
3.11.15 SYS_INFO ... 382
3.11.16 SYS_SERVER_ALIVE ... 385
3.11.17 SYS_SNMP_ACTIVE ... 386
3.11.18 SYS_USER_LANGUAGE ... 387
3.11.19 TOGGLE_SYSTEM_STATUS ... 388
3.12 Datum und Zeit ... 389
3.12.1 ADD_DAYS ... 389
3.12.2 ADD_PERIOD ... 390
3.12.3 ADD_TIME ... 392
3.12.4 ADD_TIMESTAMP ... 393
3.12.5 CALE_LOOK_AHEAD ... 394
3.12.6 CONV_DATE ... 396
3.12.7 CONV_TIMESTAMP ... 397
3.12.8 DAY_OF_YEAR ... 398
3.12.9 DIFF_DATE ... 399
3.12.10 DIFF_TIME ... 401

3.12.11 FIRST_OF_PERIOD 402
3.12.12 LAST_OF_PERIOD 404
3.12.13 SUB_DAYS 406
3.12.14 SUB_PERIOD 407
3.12.15 SUB_TIME 409
3.12.16 SUB_TIMESTAMP 410
3.12.17 SYS_DATE 411
3.12.18 SYS_DATE_PHYSICAL 413
3.12.19 SYS_LDATE 414
3.12.20 SYS_TIME 415
3.12.21 SYS_TIME_PHYSICAL 417
3.12.22 SYS_TIMESTAMP_PHYSICAL 419
3.12.23 VALID_CALE 420
3.12.24 VALID_DATE 421
3.12.25 VALID_TIME 422
3.12.26 WEEK_NR 423
3.12.27 WEEKDAY_NR 424
3.12.28 WEEKDAY_XX 425
3.12.29 YEAR_9999 426
3.13 Rechnen 427
3.13.1 ADD 427
3.13.2 DIV 429
3.13.3 GET_BIT 431
3.13.4 MOD 432
3.13.5 MULT 433
3.13.6 RANDOM 434
3.13.7 SUB 436
3.14 Strings 437
3.14.1 ARRAY_2_STRING 437
3.14.2 ALPHA2RUNNR 439
3.14.3 CONVERT 440
3.14.4 FORMAT 441
3.14.5 HEX 443
3.14.6 ISNUMERIC 444

3.14.7 RUNNR2ALPHA 445
3.14.8 STR_CAT 446
3.14.9 STR_CUT, MID, SUBSTR 446
3.14.10 STR_ENDS_WITH 448
3.14.11 STR_FIND 449
3.14.12 STR_FIND_REVERSE 450
3.14.13 STR_ISLOWER 451
3.14.14 STR_ISUPPER 452
3.14.15 STR_LC, CONV_LC 453
3.14.16 STR_LENGTH, STR_LNG 454
3.14.17 STR_LTRIM 455
3.14.18 STR_MATCH 456
3.14.19 STR_PAD 457
3.14.20 STR_REVERSE 458
3.14.21 STR_RTRIM 459
3.14.22 STR_SPLIT 460
3.14.23 STR_STARTS_WITH 461
3.14.24 STR_SUBSTITUTE 462
3.14.25 STR_SUBSTITUTE_VAR, STR_SUB_VAR 464
3.14.26 STR_TRIM 465
3.14.27 STR_UC, CONV_UC 466
3.14.28 UC_CRLF 467
4 AE JCL für Anwendungen 468
4.1 Allgemeines zur AE JCL für Anwendungen 468
4.2 Oracle Applications 468
4.2.1 Allgemeines zur Oracle Applications-JCL 468
4.2.2 OA_ADD_LAYOUT 469
Anmerkungen 469
4.2.3 OA_ADD_NOTIFICATION 470
Anmerkungen 470
4.2.4 OA_ADD_PRINTER 471
Anmerkungen 471
4.2.5 OA_SET_PRINT_DEFAULTS 472
4.2.6 OA_SUBMIT_REQUEST 473

Anmerkungen ... 473
4.3 PeopleSoft ... 474
4.3.1 Allgemeines zur PeopleSoft-JCL ... 474
4.3.2 PS_GET_HEARTBEAT ... 474
4.3.3 PS_GRANT_OUTPUT_ACCESS ... 476
4.3.4 PS_MODIFY_RUNCONTROL ... 477
4.3.5 PS_RUN_JOB ... 478
4.3.6 PS_RUN_PROCESS ... 480
4.3.7 PS_SET_BINDVAR ... 482
4.4 SAP ... 483
4.4.1 SAP Basis ... 483
Allgemeines zur SAP-JCL ... 483
R3_ACTIVATE_CM_PROFILE ... 483
Syntax ... 484
Anmerkungen ... 484
Beispiel ... 484
R3_ACTIVATE_EXT_COMMAND ... 484
Syntax ... 484
Beispiel ... 485
R3_ACTIVATE_EXT_PROGRAM ... 486
Syntax ... 486
Beispiel ... 487
R3_ACTIVATE_INTERCEPTED_JOBS ... 487
Syntax ... 487
Anmerkungen ... 490
Beispiel ... 490
R3_ACTIVATE_JOBS ... 490
Syntax ... 490
Beispiel ... 494
Syntax ... 494
Beispiel ... 498
R3_ACTIVATE_REPORT ... 498
Syntax ... 498
Anmerkungen ... 505

Beispiel 506
R3_ACTIVATE_SESSIONS 506
Syntax 506
Beschreibung 508
Beispiele 508
R3_CALL_TRANSACTION 509
Syntax 509
Anmerkungen 510
Beispiel 510
R3_COPY_VARIANT 510
Syntax 510
Beispiel 512
R3_CREATE_OUTPUT_REQUEST 512
Syntax 512
Beispiel 513
R3_CREATE_VARIANT 513
Syntax 514
Anmerkungen 514
Beispiel 515
R3_DEACTIVATE_CM_PROFILE 515
Syntax 515
Anmerkungen 515
Beispiel 516
R3_DELETE_NODE 516
Syntax 516
Anmerkungen 516
Beispiel 517
R3_DELETE_VARIANT 517
Syntax 517
Beispiel 517
R3_GET_APPLICATION_RC 518
Syntax 518
Anmerkungen 518
Beispiel 519

R3_GET_APPLICATIONLOG 519
Syntax 519
Anmerkungen 521
R3_GET_EVENT 521
Syntax 522
Anmerkungen 522
Beispiel 522
R3_GET_INTERCEPTION 522
Syntax 523
Anmerkungen 523
Beispiel 523
R3_GET_JOB_SPOOL 524
Syntax 524
Anmerkungen 526
Beispiel 527
R3_GET_JOBLOG 527
Syntax 527
Anmerkungen 528
Beispiel 528
R3_GET_JOBS 528
Syntax 528
Beschreibung 531
Beispiele 531
R3_GET_MONITOR 532
Syntax 532
Anmerkungen 533
R3_GET_SESSIONS 533
Syntax 533
Beschreibung 535
Beispiele 535
R3_GET_SPOOLREQUESTS 536
Syntax 536
Beschreibung 538
Beispiel 538

R3_GET_SYSTEMLOG 538
Syntax 538
Anmerkungen 539
Beispiel 540
R3_GET_VARIANTS 540
Syntax 540
Beispiel 541
R3_GET_VARIANT_CONTENTS 541
Syntax 541
Beispiele 542
R3_IMPORT_CALENDAR 543
Syntax 543
Beschreibung 544
Beispiel 544
R3_IMPORT_JOBS 544
Syntax 545
Beschreibung 548
Beispiel 548
R3_MODIFY_INTERCEPTION 549
Syntax 549
Anmerkungen 549
Beispiel 549
R3_MODIFY_JOB 550
Syntax 550
Beschreibung 552
Beispiel 552
R3_MODIFY_VARIANT 553
Syntax (Parameter) 553
Syntax (Select-Optionen) 554
Anmerkungen 555
Beispiel (Parameter) 556
Beispiele (Select-Optionen) 556
R3_RAISE_EVENT 556
Syntax 557

Anmerkungen 557
Beispiel 557
R3_SCHEDULE_JOB_CANCEL 557
Syntax 558
Beschreibung 558
Beispiel 558
R3_SEND_SPOOL_REQUEST 558
Syntax 558
Anmerkungen 561
Beispiel 561
R3_SET_BDCDATA 561
Syntax 562
Anmerkungen 562
Beispiel 563
R3_SET_FREE_SELECTION 563
Syntax 563
Anmerkungen 564
Beispiel 564
R3_SET_LOG_ATTR 564
Syntax 565
Anmerkungen 566
Beispiel 567
R3_SET_PERF_ATTR 567
Syntax 567
Anmerkungen 568
Beispiel 569
R3_SET_PRINT_DEFAULTS 569
Syntax 569
Anmerkungen 574
Beispiel 574
R3_SET_SELECT_OPTION 574
Syntax 575
Anmerkungen 575
Beispiel 576

R3_SET_STATUS_ATTR 576
Syntax 576
Anmerkungen 577
Beispiel 578
R3_SET_TEXT_ATTR 578
Syntax 578
Anmerkungen 578
Beispiel 579
R3_SWITCH_OPMODE 579
Syntax 579
Beispiel 580
4.4.2 SAP BCA 580
BCA_ACTIVATE_PROCESS 580
Syntax 580
Anmerkungen 581
4.4.3 SAP BW 582
Allgemeines zur SAP BW-JCL 582
Script-Sprachmittel 582
BW_ACTIVATE_CHAIN 582
Syntax 583
Beschreibung 585
Beispiel 586
BW_ACTIVATE_INFOPACKAGES 586
Syntax 586
Anmerkungen 587
Beispiele 587
BW_GET_CHAINS 588
Syntax 588
Anmerkungen 589
Beispiel 589
BW_GET_INFOPACKAGES 589
Syntax 589
Anmerkungen 589
Beispiel 590

BW_RESTART_CHAIN 590
Syntax 590
Beschreibung 592
Beispiel 593
BW_SET_CONSTRAINT 593
Syntax 593
Beschreibung 595
Beispiel 595
BW_SET_INFOPACKAGE_SELECTION 595
Syntax 595
Anmerkungen 596
Beispiel 597
4.4.4 SAP XI 597
XI_GET_CHANNEL 597
Syntax 597
Anmerkungen 598
Beispiel 599
XI_SET_CHANNEL 599
Syntax 599
Anmerkungen 600
Beispiel 601
4.5 Siebel 601
4.5.1 SI_START_TASK 601
5 AE JCL für JMX 603
5.1 Allgemeines zur JMX-JCL 603
5.2 JMX_COMPOSITE_ADD 604
5.3 JMX_CREATE_MBEAN 605
5.4 JMX_GET_AGENT 606
5.5 JMX_GET_ATTRIBUTE 607
5.6 JMX_GET_INFO 607
5.7 JMX_INVOKE 608
5.8 JMX_QUERY_NAMES 609
5.9 JMX_SET_ATTRIBUTE 610
5.10 JMX_UNREGISTER_MBEAN 610

5.11 JMX_WAIT_FOR_NOTIFICATION 611
6 AE JCL für SQL 612
6.1 Allgemeines zur SQL-JCL 612
6.2 SQL_EXECUTE_JOB 612
6.3 SQL_GET_COLUMNS 613
6.4 SQL_GET_JOBS 614
6.5 SQL_GET_TABLES 615
6.6 SQL_ON_ERROR 616
6.7 SQL_ON_ROWCOUNT_ZERO 617
6.8 SQL_SET_STATEMENT_TERMINATOR 618
Glossar 619
.1 A 619
.2 B 620
.3 C 620
.4 D 621
.5 E 621
.6 F 622
.7 G 622
.8 H 622
.9 I 623
.10 J 623
.11 K 623
.12 L 623
.13 M 624
.14 N 624
.15 O 624
.16 P 625
.17 Q 626
.18 R 626
.19 S 627
.20 T 628
.21 U 628
.22 V 629
.23 W 630

.24 X 630
.25 Z 630

1 Einführung

1.1 Über das AE Script

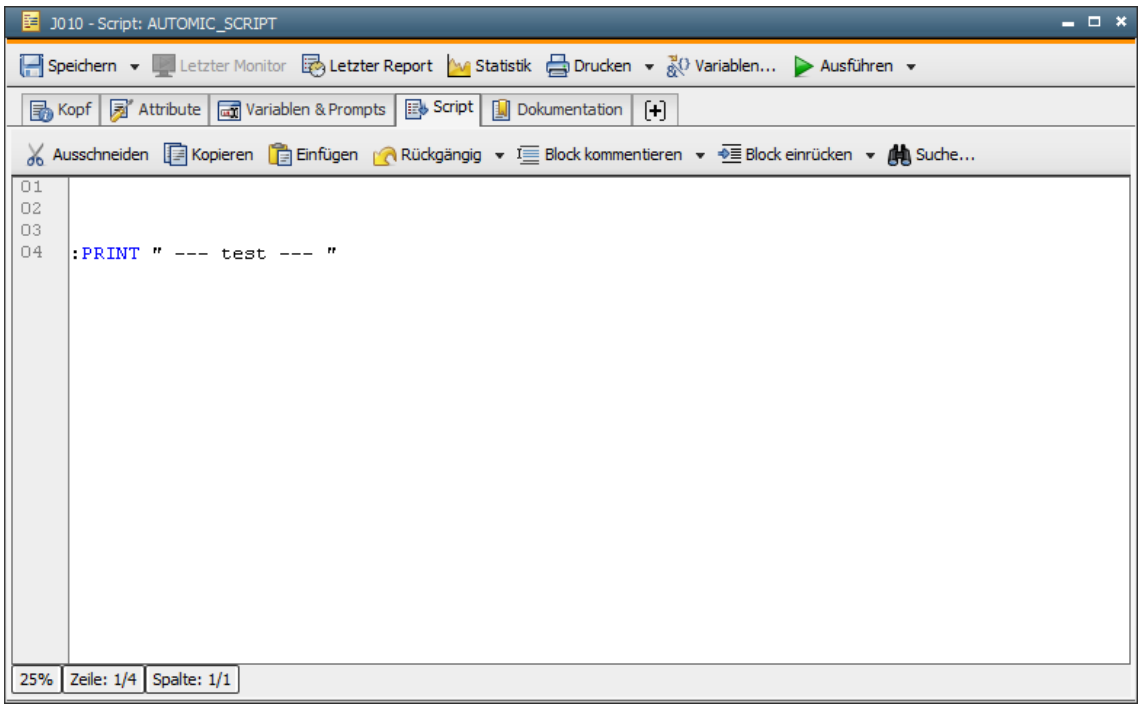
Automation Engine besitzt eine eigene umfangreiche Script-Sprache. Sie ermöglicht eine tiefgreifende Automation Ihrer Prozesse. Alle aktivierbaren Objekte weisen daher eine Script-Registerkarte und in einigen Fällen zusätzlich spezielle Registerkarten auf in der Sie Anweisungen und Funktionen festhalten können.

Dieses Einführungskapitel soll dazu dienen Ihnen die Grundlagen der AE Script-Sprache näherzubringen. Schritt für Schritt werden Sie mit Teilaspekten wie, z.B. dem Aufbau des Scriptes und der Definition von Variablen, vertraut gemacht. Am Ende des Kapitels sind Tipps aufgelistet, die Ihnen bei der Erstellung Ihrer AE Scripts behilflich sind. Doch nicht nur Script-Neulinge werden unterstützt. Auch Experten finden schnell gesuchte Informationen zu den Script-Sprachmitteln, da bei jedem die Syntax mit den Parameterwerten in übersichtlicher Weise vermerkt ist. Ein rasches Nachschlagen wird dadurch ermöglicht.

Beachten Sie, dass das Berechtigungssystem von der Automation Engine auch die Anwendung des AE Scripts miteinschließt. Besitzt ein Benutzer beispielsweise keine Rechte auf Variablen-Objekte, so kann er diese auch nicht mit den entsprechenden Script-Funktionen auslesen bzw. befüllen.

Beachten Sie folgendes beim Aktivieren oder Wiederanlauf von Objekten via UserInterface: Wird das UserInterface beendet, bevor die Script-Generierung abgeschlossen ist, so kommt es gegebenenfalls nicht zum gewünschten Ergebnis.

Wir wünschen Ihnen viel Freude beim Erlernen und Einsetzen der AE Script-Sprache!



1.2 Erste Schritte

1.2.1 Einführung

Dieses Kapitel bringt Ihnen mit Hilfe verschiedener Lektionen, welche Sie Schritt-für-Schritt abarbeiten können, den Umgang mit der AE-Scripsprache näher und ermöglicht Ihnen so einen einfachen Einstieg.

Mit der Scriptsprache von AE ist es möglich, verschiedene Abläufe zu steuern und Verarbeitungen durchzuführen. Sie können beispielsweise Objekte ändern und ausführen oder Rechenoperationen durchführen.
Scripts vereinfachen einerseits Prozesse, da sie viele manuellen Schritte ersetzen. Andererseits stellen sie einzigartige Funktionalitäten bereit, wie z.B.: Rechnen-Funktionen oder Prozess-Schleifen.

Alle aktivierbaren Objekte können Scripts enthalten. Dazu dienen die Script-Registerkarten.

Eine grobe Beschreibung des allgemeinen Script-Aufbaus finden Sie hier Scriptaufbau: Script-Aufbau.

1.2.2 Vorab-Checkliste

Bestimmte Scriptzeilen und Sprachmittel erfordern spezielle Vorraussetzungen.

Berechtigungen

Scripts werden beim Ausführen von Objekten prozessiert. Der Benutzer benötigt daher die Berechtigung "X" für die zu startenden Objekte.

JCL

JCL ist die Abkürzung für "Job Control Language". Dabei handelt es sich um Kommandos und Befehle, die auf bestimmten Rechnern ausgeführt werden, zum Beispiel ein "dir" Kommando auf einem Windows-Rechner.

Neben Betriebssystem-Befehlen stehen für bestimmte Plattformen bzw. Anwendungen (zum Beispiel: SAP) spezielle AE JCL-Sprachmittel zur Verfügung. Diese finden Sie in folgenden Kapiteln:

- AE JCL für Anwendungen
- AE JCL für JMX
- AE JCL für SQL

Voraussetzungen für die Verwendung von JCL ist ein Job-Objekt und ein Agent für das jeweilige Betriebssystem (z.B.: Windows), die Plattform (z.B.: SAP) oder die Anwendung (z.B.: SQL). Der Agent muss aktiv sein und am Automation Engine System angemeldet sein. Weiters muss der Agent für den jeweiligen Mandanten freigeschalten sein (siehe: Agent - Registerkarte **Berechtigungen**).
Damit die JCL-Befehle ausgeführt werden können, ist auch ein Login-Objekt erforderlich, welches

gültige Login-Informationen für das Betriebssystem, die Plattform bzw. die Anwendung enthält. Dieses ist im Job-Objekt auszuwählen.

JCL-Befehle können nur in der Registerkarte **Script** von Jobs verwendet werden!

Mit dem Sprachmittel :DATA werden JCL-Zeile explizit als solche deklariert.

Spezielle AE-Sprachmittel

Folgende Sprachmittel erfordern individuelle Voraussetzungen:

- :REGISTER_OUTPUTFILE
  Verwendung nur in der Registerkarte **Script** von UNIX- und Windows-Jobs möglich.
- SEND_MAIL
  E-Mail Kopplung muss konfiguriert sein
- :SEND_SNMP_TRAP
  SNMP ist konfiguriert
- SYS_ACT_HOST
  Verwendung in der Script-Registerkarte von Jobs oder Dateisystem bzw. Konsolen-Ereignissen (in diesem Fall auch !Script möglich).
- PREP_PROCESS_FILE und PREP_PROCESS_FILENAME
  Betriebssystem-Agent
- PREP_PROCESS
  Agent entsprechend Sprachmittel-Beschreibung
- GET_CONSOLE, GET_FILESYSTEM und GET_WIN_EVENT
  Verwendung nur in Ereignis-Objekten möglich
- ILM
  Verwendung von ILM

1.2.3 Das erste Script

Ziel:
- Text mittels Script in das Aktivierungsprotokoll schreiben

Lektion 1

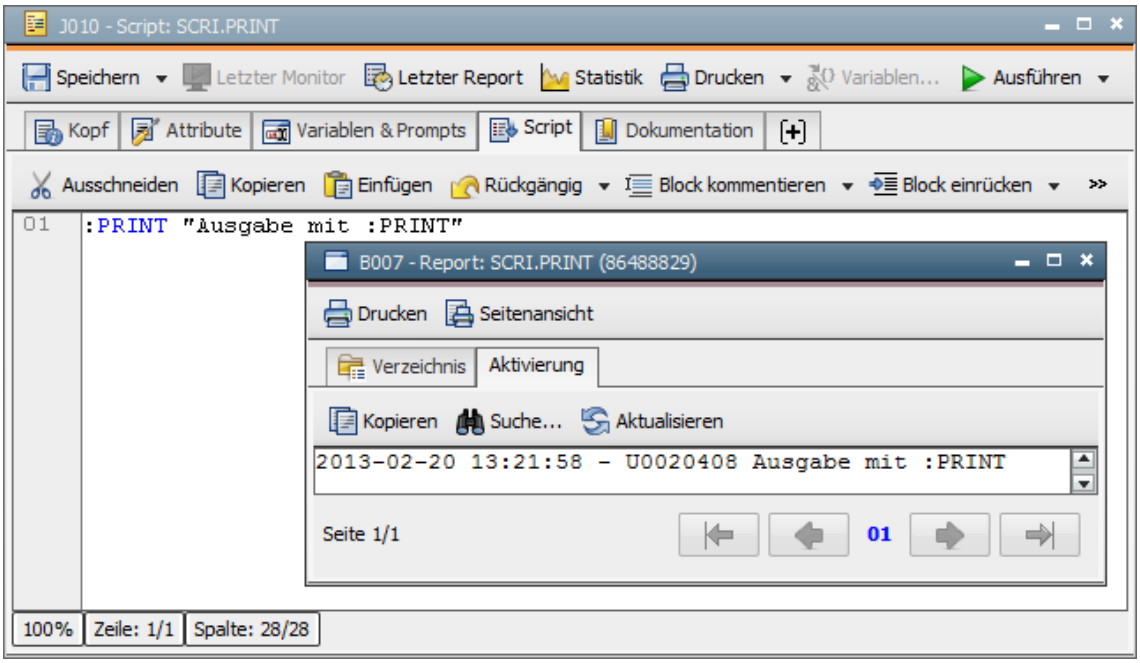
Legen Sie zuerst ein ausführbares Objekt an, am besten ein Script-Objekt (SCRI). Öffnen Sie dieses und wechseln Sie in die Registerkarte **Script**.

Hier können Sie Script-Zeilen einfügen, die beim Ausführen des Objektes abgearbeitet werden.

Um Informationen in das Aktivierungsprotokoll (Report) der Aufgabe zu schreiben, dient das Script-Sprachmittel :PRINT. Tragen Sie folgende Zeile in die Script-Registerkarte ein:

```
:PRINT "Ausgabe mit :PRINT"
```

Speichern Sie das Script-Objekt und starten Sie es. Öffnen Sie anschließend den letzten Report. In der Registerkarte "Aktivierung" sehen Sie den gewünschten Text mit führendem Zeitstempel (Zeitpunkt der Ausgabe).



1.2.4 Script-Anweisungen

Ziele:
- Script-Anweisungen verwenden

Lektion 2

Jede Scriptzeile in der AE, bei der es sich nicht um JCL oder ein Kommentar handelt, beginnt mit einer Script-Anweisung.

Script-Anweisungen sind AE Sprachmittel, die mit einem Doppelpunkt **:** beginnen und blau im Script-Editor des UserInterface hervorgehoben werden. Sie stehen oft im Zusammenhang mit einer Wertzuweisung.

Beim Sprachmittel :PRINT handelt es sich beispielsweise um eine Script-Anweisung (siehe Lektion 1).

Ein weiteres Beispiel für eine Anweisung, ist das Sprachmittel :SEND_MSG, welches eine Nachricht an einen bestimmten AE-Benutzer sendet. Die Nachricht wird im Meldungsfenster angezeigt. Beispiel:

```
: SEND_MSG "HUBER","EDV","Bitte Sicherung starten!"
```

Mit der Script-Anweisung :PUT_ATT können Sie die Attribute jenes Objektes ändern, in welchem das Sprachmittel verwendet wird. Folgende Scriptzeile setzt beispielsweise das Attribut "Kostenstelle" (Registerkarte **Attribute** z.B.: beim Script-Objekt):

```
: PUT_ATT INT_ACCOUNT="Kostenstelle"
```

1.2.5 Script-Variablen

Ziele:
- Script-Variablen erstellen
- Variablen mit Werte befüllen
- Variablen-Werte ausgeben

Lektion 3

Script-Variablen dienen zur Speicherung und Übergabe von Werten im Script. Sie können beispielsweise den Rückgabewert von Script-Funktionen erhalten.

Script-Variablen können mit einem bestimmten Datentyp erstellt werden. Der Datentyp legt fest, welche Werte in Script-Variablen gespeichert werden dürfen. Folgende verschiedene Datentypen stehen zur Verfügung:

- **unsigned** - Positive Ganzzahlen
- **signed** - Negative und positive Ganzzahlen
- **float** - Positive und negative Gleitkommazahlen
- **string** - Zeichenkette, Wert muss unter Hochkomma stehen (Script-Literal)

Der Name von Variablen beginnt mit dem Zeichen **&**, das zweite Zeichen darf dabei keine Zahl sein.

In den folgenden Schritten werden wir 2 Script-Variablen erstellen, Wertzuweisungen vornehmen und die Werte im Aktivierungsprotokoll ausgeben:

Öffnen Sie zuerst ein beliebiges, aktivierbares Objekt und wechseln Sie in eine Script-Registerkarte.

Die erste Script-Variable erstellen wir mit dem Sprachmittel :DEFINE. Der Datentyp wird dabei fix gesetzt und ist nachträglich nicht mehr änderbar. Da die Variable eine negative Zahl speichern soll, wird der Datentyp "signed" gewählt. Die Wertzuweisung nehmen wir mit dem Sprachmittel :SET vor.

```
:DEFINE &zahl#,signed
:SET &zahl# = -1
```

Die zweite Script-Variable erstellen wir direkt mit :SET erstellt und setzen sie auf den Wert "test". In diesem Fall kann die Variable nachträglich auf jeden beliebigen Wert geändert werden (d.h.: der Datentyp richtet sich nach dem Wert).

```
:SET &string# = "test"
```

Nutzen Sie die folgende Scriptzeile um die Werte beider Variablen in das Aktivierungsprotokoll schreiben. Um den Variablennamen auszugeben ist das Zeichen & doppelt anzugeben.

```
:PRINT "&&zahl# = &zahl#, &&string# = &string#"
```

Nach Speichern und Ausführen des Objektes finden Sie folgende Zeile im Aktivierungsprotokoll:

```
2011-04-06 14:48:17 - U0020408 &zahl# = -0000000000000001, &string# =
test
```

Weiterführende Informationen zu Script-Variablen finden Sie in dem entsprechenden Dokument.

1.2.6 Script-Funktionen

Ziele:
- Script-Funktion aufrufen
- Rückgabewert der Funktion in eine Script-Variable speichern
- Variable im Aktivierungsprotokoll ausgeben

Lektion 4

Script-Funktionen sind AE-Sprachmittel, die Rückgabewerte liefern. Viele Funktionen liefern nicht nur Werte, sondern führen auch bestimmte Aktionen aus. Script-Funktionen dürfen nicht am Zeilenbeginn stehen und können somit nur im Zusammenhang mit Script-Anweisungen verwendet werden. Sie werden im Script-Editor des UserInterface rot hervorgehoben.
Der Funktions-Name ist gefolgt von Klammern ( ), die Parameter enthalten können.

In folgender Lektion werden wir eine Rechenoperation mit Hilfe einer Script-Funktionen durchführen.

Zuerst erstellen wir eine Script-Variable mit dem Datentyp "float" (siehe auch Lektion 3).

```
:DEFINE &ergebnis#, float
```

Danach verwenden wir die Script-Funktion DIV um eine Division durchzuführen. Die Funktion besitzt 2 Parameter: Zahl 1 und 2.
Zahl 1 wird dabei durch Zahl 2 dividiert. Rückgabewert der Funktion ist das Ergebnis der Divison. Dieses speichern wir in eine Script-Variable, wodurch die Script-Anweisung :SET notwendig ist.

Wenn wir nun Script-Anweisung :SET, Script-Variable &ergebnis# und Script-Funktion DIV kombinieren, erhalten wir folgende Zeile:

```
:SET &ergebnis# = DIV(1,4)
```

Nun wollen wir das Ergebnis im Aktivierungsprotokoll ausgeben.

```
:PRINT "Ergebnis: 1/4 = &ergebnis#"
```

Ergebnis im Aktivierungs-Report:

```
2011-04-06 14:48:17 - U0020408 Ergebnis: 1/4 =
+0000000000000000.2500000000000000
```

Ein weiteres Beispiel für eine Script-Funktion ist FORMAT, mit der Sie nun das Ergebnis der Division formatieren können. In unserem Fall entfernen wir unnötige führende und abschließende Nullen.

```
:SET &format# = FORMAT(&ergebnis#,"0.00")
:PRINT "Ergebnis formatiert: &format#"
```

Aktivierungsprotokoll:

```
2011-04-06 14:48:17 - U0020408 Ergebnis formatiert:0.25
```

1.2.7 Verwendung von JCL

Ziele:
- Ausführen von Betriebssystem-Kommandos

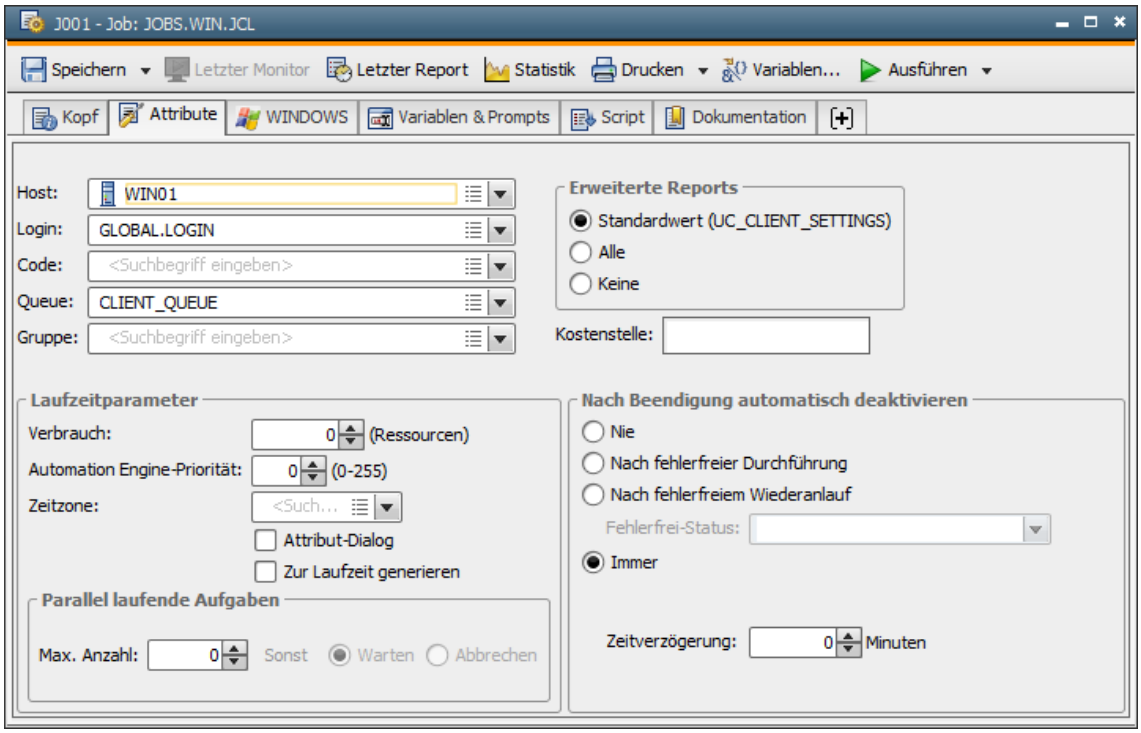
Lektion 5

JCL (=Job Control Language) kann nur in der Registerkarte **Script** von Jobs verwendet werden. Dabei handelt es sich um Befehle die nicht auf der Seite der Automation Engine, sondern auf einem bestimmten Betriebssystem, Plattform, Anwendung ausgeführt werden.

Mit dem AE-Sprachmittel :DATA können JCL-Zeilen explizit als solche angegeben werden. JCL wird im Script-Editor braun gekennzeichnet.

Folgende Lektion soll Ihnen den Umgang mit JCL näherbringen.

Zuerst wollen wir ein Betriebssystem-Kommando auf einem Rechner ausführen. Legen Sie dazu ein Job-Objekt an, in unserem Fall ein Windows-Job. Öffnen Sie den Job und tragen Sie in der Registerkarte **Attribute** einen Agenten und ein gültiges Login-Objekt ein.

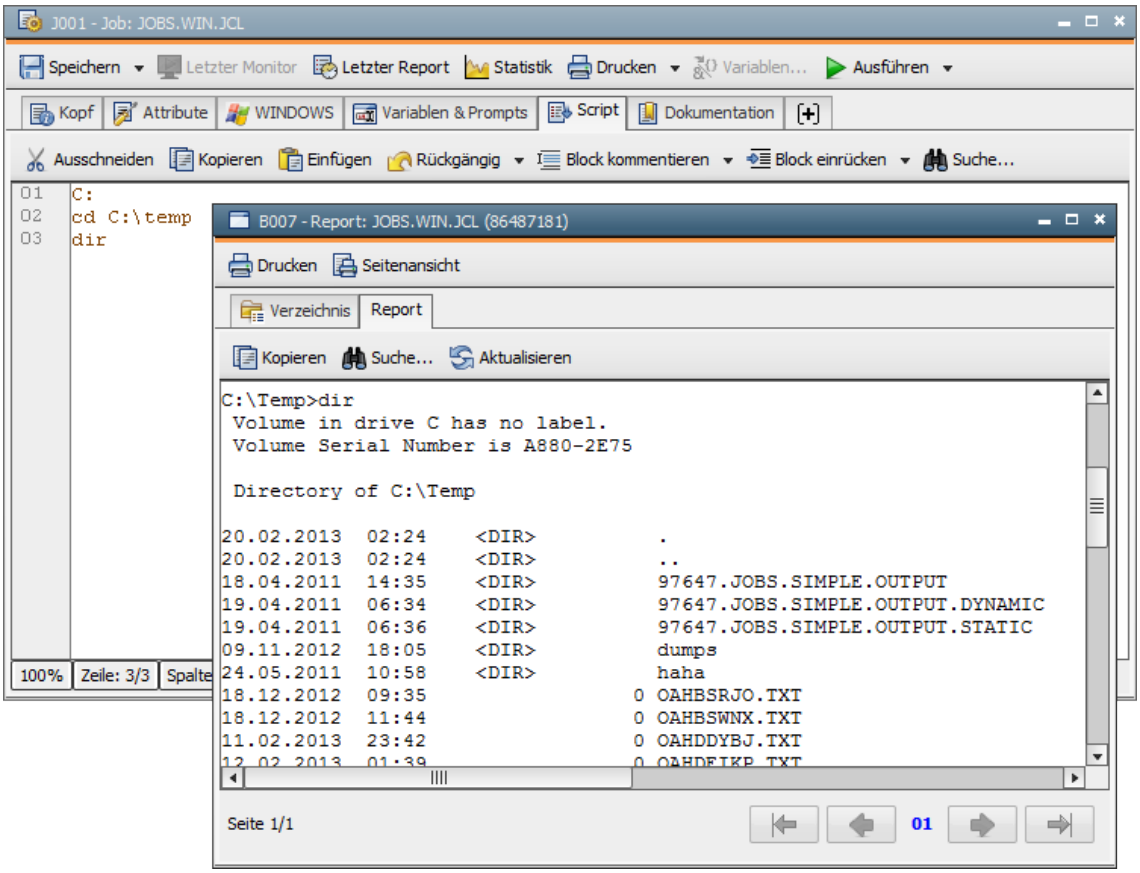


Achten Sie darauf, dass der Agent aktiv ist und dass die entsprechenden Berechtigungen vorhanden sind (sowohl die Mandanten-Berechtigungen im Agenten als auch jene im Benutzer-Objekt).

Wechseln Sie danach in die Registerkarte **Script** und tragen Sie einige Kommandos ein. Wir verwenden folgende Zeilen, um die Dateiliste eines bestimmten Verzeichnisses auf den Windows-Rechner abzufragen:

```
C:
cd C:\temp
dir
```

Speichern und starten Sie den Job. Das Ergebnis ist im Report-Dialog - Report verfügbar (Registerkarte "Report").



1.2.8 Benutzerdefinierte Dialoge

Ziele:
- Ausgabe von Text in einem Dialog-Fenster
- Abfrage von Benutzer-Werten

Lektion 6

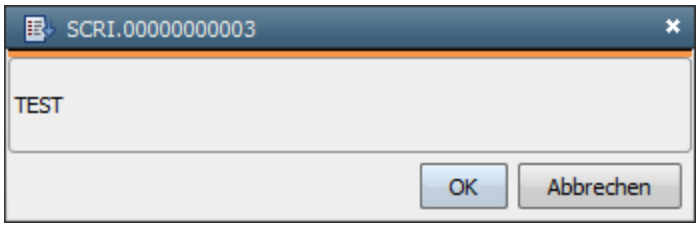
Mit der AE Scriptsprache haben Sie die Möglichkeit benutzerdefinierte Dialoge zu erstellen. Dabei können Texte ausgegeben und/oder Werte vom Benutzer abgefragt werden. Voraussetzung für die Anzeige der Dialoge ist, dass das jeweilige Objekt über eine Benutzeroberfläche (UserInterface) aktiviert wird.

Erstellen wir im ersten Schritt einen einfachen Dialog, der nur Text ausgibt. Dazu sind die Anweisungen :BEGINREAD... :ENDREAD (gibt den Beginn und Ende eines Dialoges-Fensters an) im Zusammenhang mit :PRINT zu verwenden.

Mit folgende Scriptzeile:

```
:BEGINREAD
: PRINT "TEST"
:ENDREAD
```

erhalten Sie folgendes Fenster als Ergebnis:

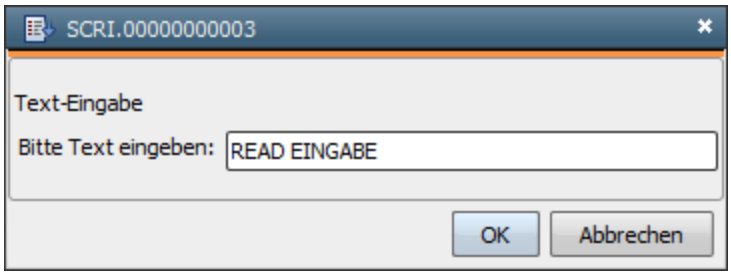


Im nächsten Schritt erstellen wir einen Dialog, der eine Eingabe ermöglicht. Dazu dient das Sprachmittel :READ (oft wird auch die Bezeichnung READ-Masken verwendet).

In folgendem Beispiel wird eine :PRINT-Ausgabe und :READ-Eingabe in einem Dialog kombiniert. Der Wert wird in eine Script-Variable gespeichert (in diesem Fall wird die Variable direkt mit dem Sprachmittel :READ erstellt). Danach erfolgt wird die Variable im Aktivierungsprotokoll ausgegeben.

```
:BEGINREAD
: PRINT "Text-Eingabe"
: READ &EINGABE#,"OO","Bitte Text eingeben"
:ENDREAD

:PRINT "Benutzer-Text: &EINGABE#"
```



1.2.9 IF-Bedingungen

Ziele:
- IF-Bedingung erstellen
- ELSE-Block hinzufügen

Lektion 7

Im AE Script ist es auch möglich, Anweisungen erst dann auszuführen, wenn bestimmte Bedingungen erfüllt sind. Bedingungen werden mit dem Sprachmittel :IF... :ELSE... :ENDIF erstellt.

Zu Beginn erstellen wir eine einfache Bedingung für den Vergleich von 2 Zahlen. Als erstes ist das Sprachmittel :IF mit einer Bedingung anzugeben. Tragen Sie danach alle Scriptzeilen ein, die beim eintreffenden der Bedingung durchgeführt werden sollen. Jeder IF-Block wird mit :ENDIF abgeschlossen.

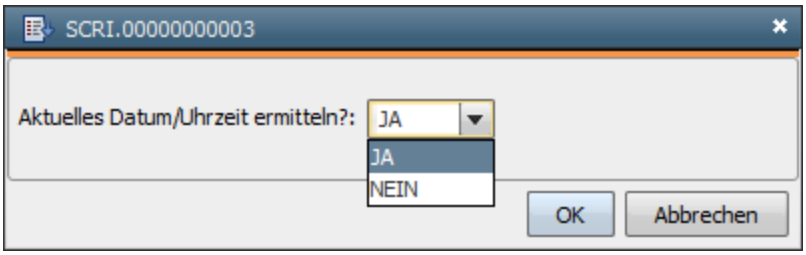
```
:IF 1<2
: PRINT "Bedingung trifft zu"
:ENDIF
```

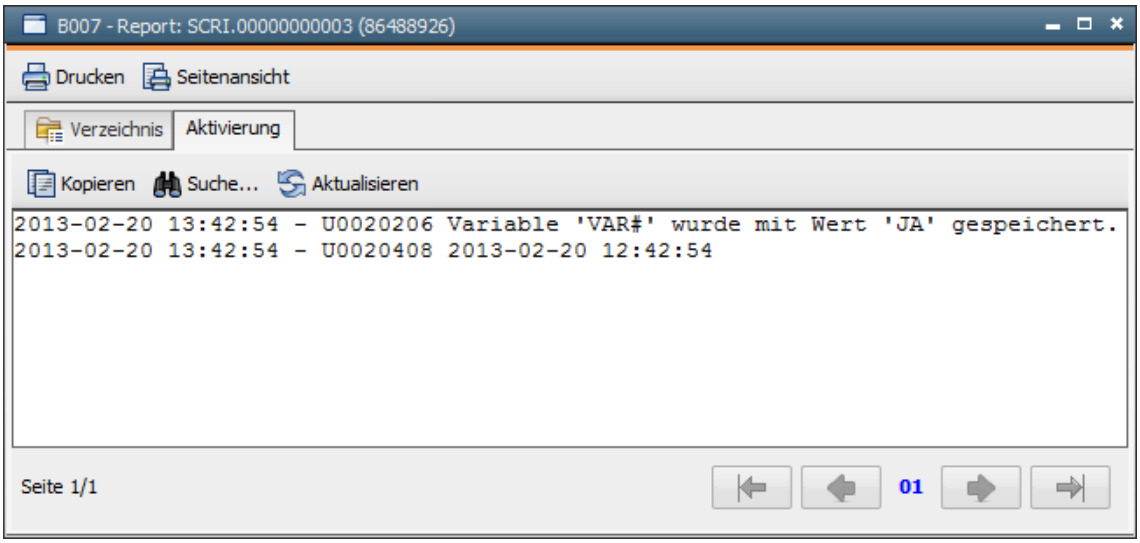
Im nächsten Beispiel kombinieren wir eine READ-Maske (siehe vorige Lektion) mit einer IF-Anweisung. Zuerst erstellen wir eine Maske, bei welcher der Benutzer zwischen den Werten "JA" und "NEIN" auswählen kann. Wählt der Benutzer "Ja", wo wird mit Hilfe einer Script-Funktion, das aktuelle Datum und Uhrzeit ermittelt und im Aktivierungsprotokoll ausgegeben.

```
:READ &VAR#,"'JA','NEIN'", "Aktuelles Datum/Uhrzeit ermitteln?","JA"

:IF &VAR# = "JA"
: SET &ZEIT# = SYS_TIMESTAMP_PHYSICAL()
: PRINT &ZEIT#
:ELSE
: PRINT "Keine Ermittlung von Uhrzeit/Datum"
:ENDIF
```

Mit :ELSE innerhalb eines IF-Blocks bestimmen Sie die Anweisungen, die durchgeführt werden sollen, wenn die Bedingung nicht zutrifft. In diesem Fall wird eine Meldung in den Aktivierungsreport geschrieben.





1.2.10 Schleifen im Script

Ziele:
- WHILE-Schleifen
- PROCESS-Schleifen

Lektion 8

Schleifen führen bestimmte Script-Zeilen mehrmals durch. Im AE Script stehen Ihnen 2 unterschiedliche Schleifen zur Verfügung:

- WHILE-Schleifen: Sprachmittel :WHILE...:ENDWHILE
- PROCESS-Schleifen: Sprachmittel :PROCESS... :TERM_PROCESS... :ENDPROCESS

While-Schleifen wiederholen einen bestimmten Block solange eine bestimmte Bedingung erfüllt ist. Die Wiederholung endet erst dann, wenn die Bedingung nicht mehr zutrifft.

⚠ Beachten Sie, dass es bei While-Schleifen, welche sehr oft hintereinander durchlaufen werden, zu einem Abbruch der Script-Prozessierung kommt. Durch dieses Verhalten werden Endlos-Schleifen vermieden.

Der Prozess-Schleife wird eine Referenz auf eine Datensequenz übergeben. Die Anzahl der Schleifen-Durchläufe ergibt sich dabei aus der Zeilen-Anzahl der Datensequenz. Bei einer Datensequenz handelt es sich um eine interne Auflistung, zum Beispiel: eine Liste von Dateien oder die Einträge von Variablen-Objekten. Welche Werte in der Datensequenz stehen, hängt vom jeweiligen Sprachmittel ab.

Zuerst werden wir anhand eines Beispiels die Verwendung von While-Schleifen erklären.

Aufgabe des Scripts ist zu überprüfen, ob ein bestimmter Agent verfügbar ist. Erst wenn dieser aktiv ist, soll ein Job auf diesem Agenten gestartet werden. Das Script kann sich dabei in einem beliebigen ausführbaren Objekt befinden (z.B.: SCRI). Voraussetzung dafür ist somit ein Agent, auf welchem der Benutzer Jobs ausführen darf.

Verwenden Sie im ersten Schritt das Sprachmittel SYS_HOST_ALIVE, um zu überprüfen ob der Agent aktiv ist. In unserem Beispiel wird der Agent WIN03 gewählt. Liefert das Sprachmittel den Rückgabewert "Y", so ist Agent verfügbar.

```
:SET &AGENT# = WIN03
:SET &ALIVE# = SYS_HOST_ALIVE(&AGENT#)
```

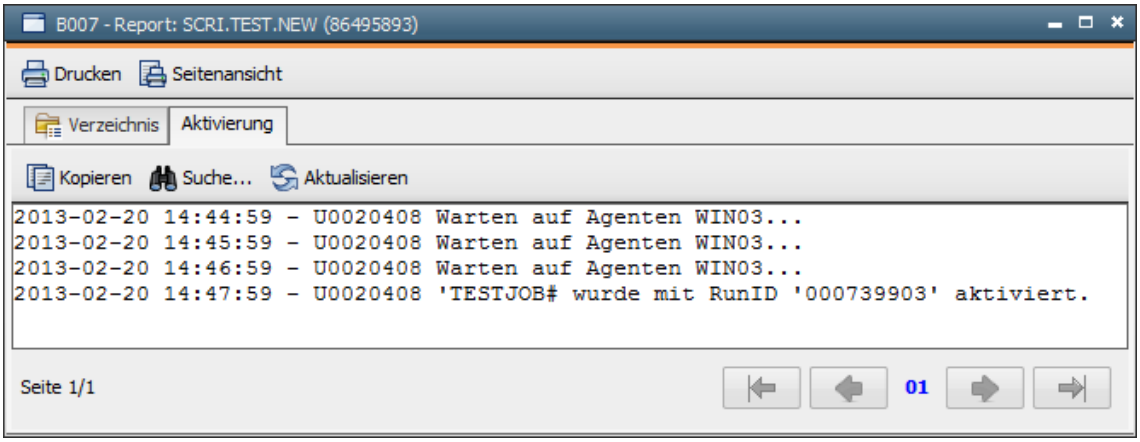
Als nächstes ist eine While-Schleife zu erstellen. Diese soll die Prüfung des Agenten mit einer gewissen Zeitverzögerung (um zuviele Schleifendurchläufe zu vermeiden) wiederholen.
Enden soll die Schleife dann, wenn der Agent aktiv ist. Die Bedingung der Schleife muss daher lauten: &ALIVE# NE "Y". Der Vergleichsoperator "NE" bedeutet "Entspricht nicht". Andere Vergleichs-Operatoren finden Sie in der Beschreibung des Sprachmittels :WHILE.

```
:WHILE &ALIVE# NE "Y"
: PRINT "Warten auf Agent &AGENT#..."
: WAIT 60
: SET &ALIVE# = SYS_HOST_ALIVE(&AGENT#)
:ENDWHILE

:SET &ACT# = ACTIVATE_UC_OBJECT(TESTJOB)
```

Ist der Agent nicht aktiv, so werden die Anweisungen der Schleife wiederholt. Durch die Anweisung :WAIT wird die Script-Prozessierung um eine gewisse Zeit verzögert, in unserem Fall wird 60 Sekunden gewartet.

Wenn der Agent wieder verfügbar ist, liefert die Prüfung (SYS_HOST_ALIVE) den Wert "Y". Die Schleife endet danach und die Scriptzeile wird erreicht, die den Job "TESTJOB" aktiviert (Script-Funktion: ACTIVATE_UC_OBJECT). Im Aktivierungsprotokoll sehen Sie, ob noch auf den Agenten gewartet wird oder ob der Job bereits aktiviert wurde.



Folgende Schritte sollen nun den Umgang mit Process-Schleifen demonstrieren.

Ziel ist es, eine Dateiliste von einem Rechner zu holen und im Aktivierungsprotokoll auszugeben.

Dazu benötigen wir das Sprachmittel PREP_PROCESS_FILENAME, welches die Dateiliste vom Rechner eines Agenten holt. Mit Hilfe von Filterbegriffen können Sie dabei nur Dateien mit bestimmten Namen auswählen. Die Dateiliste wird als Datensequenz im Script zur Verfügung gestellt.

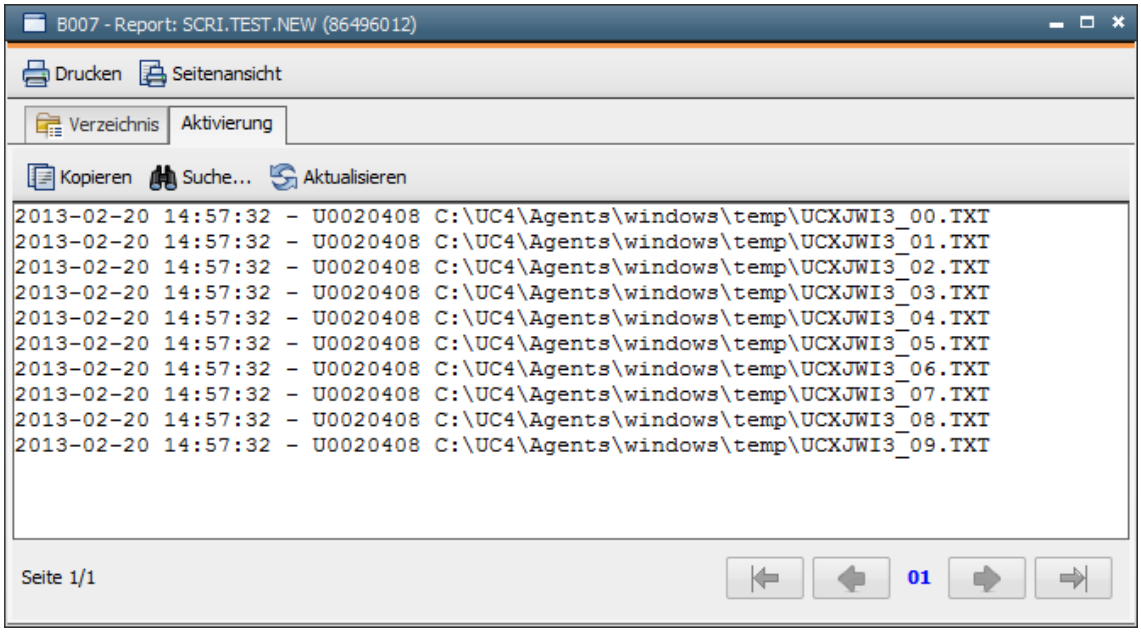
Um die Datensequenz zeilenweise abzuarbeiten, ist die Prozess-Schleife nötig. Diese wird mit :PROCESS begonnen und mit :ENDPROCESS beendet. Für jede Zeile in der Datensequenz wird die Schleife einmal durchlaufen.

Mit dem Sprachmittel GET_PROCESS_LINE erhalten Sie den Inhalt der aktuellen Datensequenz-Zeile, in unserem Fall Pfad und Namen der Datei. Diesen geben wir im Aktivierungsreport aus.

In unserem Beispiel verwenden wir den Agenten WIN01 um die Dateiliste zu holen. Wir ermitteln alle Textdateien des Verzeichnisses "C:\AUTOMIC\Agents\WIN01\temp", welches die Log-Dateien des Agenten enthält.

```
:SET&HND# = PREP_PROCESS_FILENAME("WIN01",
"C:\AUTOMIC\Agents\WIN01\temp\*.txt","Y",,)
:PROCESS&HND#
:    SET&LINE#=GET_PROCESS_LINE(&HND#)
:    PRINT&LINE#
:ENDPROCESS
```

Als Ergebnis erhalten wird Pfad und Namen aller gewählten Dateien im Aktivierungsprotokoll.



1.3 Grundlagen

1.3.1 Scriptaufbau

Ein Script besteht aus 3 verschiedenen Arten von Zeilen:

Kommentare

- Jede Zeile, die mit einem Ausrufungszeichen "!" beginnt, wird als Kommentar behandelt. Diese Zeilen werden nicht als Verarbeitungsschritte betrachtet und daher bei der Ausführung übersprungen.

  Beispiel:

  ```
  !Kommentarzeile
  ```

- Tritt das Ausrufezeichen innerhalb einer Zeile auf, so ist jene kein Kommentar.
- Mehrzeilige Kommentare können erstellt werden indem Sie die entsprechenden Zeilen markieren und dann die Schaltfläche in der Symbolleiste des UserInterfaces drücken.
- Verwenden Sie Kommentare ausgiebig, damit andere Benutzer und natürlich Sie selbst auch zu einem späteren Zeitpunkt leicht nachvollziehen können, was die Scriptzeilen bewirken.

AE Script

- Zeilen, die mit einem Doppelpunkt ":" beginnen, enthalten AE Script-Sprachmittel. Letztere werden in Script-Funktionen (liefern einen Rückgabewert) und Script-Anweisungen (liefern keinen Rückgabewert) unterteilt.

  Beispiel einer Anweisung: `:PRINT "Automation Engine"`

Beispiel einer Funktion: `:SET &ERGEBNIS# =ADD(2,2)`

- Der Unterstrich "_" kann als Fortsetzungszeichen bei überlangen Zeilen verwendet werden. Er gibt als letztes Zeichen einer Zeile an, dass die nächste Zeile fortgesetzt wird. Die Folgezeile muss mit einem Doppelpunkt beginnen.

DATA-Zeilen

- Beginnt die Zeile weder mit einem "!" noch mit einem ":" so wird sie als sogenannte DATA-Zeile behandelt. DATA-Zeilen können nur im Objekttyp "Job" angewendet werden. Sie enthalten die JCL (Job Control Language) des Zielsystems. Beginnt die DATA-Zeile mit einem ":" so muss sie mit dem Script-Sprachmittel :DATA explizit als solche deklariert werden.

  Beispiel:

  ```
  copy test.txt c:\temp
  ```

- Eine Besonderheit bilden Jobs für Enterprise Business Solutions (SAP, PeopleSoft, Oracle Applications). Für sie stellt die Automation Engine eine eigene JCL zur Verfügung.
- Die in DATA-Zeilen enthaltenen Script-Variablen werden durch ihren Wert ersetzt. Script-Variablen beginnen mit dem Sonderzeichen "&". Kommt ein "&" in einer DATA-Zeile vor, welches erhalten bleiben soll, so ist es zu verdoppeln. Kommt ein "&" vor und es folgt kein gültiger Variablenname, so bleibt die DATA-Zeile ebenfalls ungeändert.

1.3.2 Kontrollstrukturen

Das Script wird in der AE zeilenweise abgearbeitet. Da es aber in vielen Fällen notwendig ist Verarbeitungsschritte von Bedingungen abhängig zu machen, gibt es zwei Kontrollstrukturen mit denen Sie den Ablauf des Scriptes steuern können.

Die :IF-Anweisung führt Scriptzeilen nur dann durch, wenn die vorgegebene Bedingung zutrifft.

:IF Bedingung

Scriptzeilen

:ENDIF

Die :WHILE-Anweisung wiederholt Scriptzeilen solange die Bedingung zutrifft.

:WHILEBedingung

Scriptzeilen

:ENDWHILE

Nähere Informationen erhalten Sie bei der Beschreibung zu diesen Kontrollstrukturen.

1.3.3 Script-Variable

In den Scripts der Automation Engine können Sie Script-Variablen verwenden, um darin Werte zu speichern. Sie können sowohl Zahlen, Zeichenfolgen als auch Datums- und Zeitformate enthalten.

Die Deklaration einer Script-Variablen mit einem bestimmten Datentyp erfolgt mit dem Scriptsprachmittel :DEFINE. Der Datentyp legt dabei fest, welche Werte die Variable enthalten soll. Die Wertzuweisung und -änderung innerhalb des Scripts erfolgt mit der :SET-Anweisung.

```
:DEFINE &DATEI#, string
:SET &DATEI# = "temp.txt"
```

Es ist auch möglich Script-Variablen direkt mit :SET zu erstellen (ohne voriges :DEFINE). Die Variable besitzt in diesem Fall keinen spezifischen Datentyp, kann jedoch nur Zeichenketten und positive Ganzzahlen speichern.
Beispiel:

```
:SET &VAR# = 10
:SET &VAR# = "string"
```

Die Gültigkeit einer Script-Variable endet mit der Abarbeitung des Scripts.

Datentypen

Für die Verwendung von Script-Variablen gibt es vier verschiedene Datentypen mit unterschiedlichen Wertebereichen. Die Festlegung des Datentyps ist bei der Variablendeklaration mit der Script-Anweisung :DEFINE durchzuführen.

Datentyp	Beschreibung	Wertebereich
unsigned	Positive Ganzzahlen ohne Vorzeichen	0 bis +9 999 999 999 999 999
signed	Ganzzahlen mit Vorzeichen	-9 999 999 999 999 999 bis +9 999 999 999 999 999
string	Zeichenkette	1 bis 1024 Zeichen
float	Gleitkommazahlen mit Vorzeichen	-9 999 999 999 999 999.9999999999999999 bis +9 999 999 999 999 999.9999999999999999

Führen Sie zuerst die Variablendeklaration durch und nehmen Sie anschließend die passende Wertzuweisung vor.

```
:DEFINE &STRING#, string
:DEFINE &SIGNED#, signed
:DEFINE &UNSIGNED#, unsigned
:DEFINE &FLOAT#, float

:SET &STRING# = "1234abc"
:SET &SIGNED# = -5
:SET &UNSIGNED# = 24
:SET &FLOAT# = -0.50
```

Der Datentyp "float" umfasst auch positive und negative Ganzzahlen. Weiters ist es möglich, positive Ganzzahlen für den Datentyp "signed" zu verwenden.

⚠ Bereits verwendete Variablen können nicht erneut deklariert werden. Der Datentyp einer Variable ist nicht änderbar.

⚠ Beachten Sie, dass es bei Rechenoperationen mit Gleitkommazahlen zu einem ungenauen Ergebnis kommen kann!

Syntax

Der Name einer Script-Variablen besteht aus maximal 32 alphanumerischen Zeichen einschließlich der Sonderzeichen "$", "_", "@", "§" und "#". Umlaute sind nicht erlaubt. Das erste Zeichen des Namens darf keine Zahl sein. Im Script sind Variablen immer mit dem Kennzeichen "&" vor dem Namen anzugeben!

ℹ Von der Beschränkung auf 32 Zeichen sind die Script Sprachmittel :PSET und :RSET ausgenommen, für sie gilt eine Beschränkung auf 31 alphanumerische Zeichen.

⚠ Beachten Sie, dass sich der Name der Script-Variablen von vordefinierte Variablen unterscheiden muss. Es wird generell empfohlen '$' nicht an erster Stelle (direkt nach dem Zeichen '&') im Variablennamen zu verwenden.

Namen von Script-Variablen sind case-insensitiv, das heißt, es wird nicht auf die Groß- und Kleinschreibung der enthaltenen Buchstaben geachtet. Script-Variablen werden ohne führendes und abschließendes Hochkomma angegeben.

Der Name einer Script-Variablen darf nicht dem Beginn des Namens einer anderen Script-Variablen entsprechen.

Derartige Benennungen sind nur dann möglich, wenn am Ende des Namens beispielsweise ein Sonderzeichen eingefügt wird. Im folgenden Beispiel wird die Bezeichnung "Abschlussdatum" im Aktivierungsprotokoll ausgegeben.

```
:SET &AKTIVITAET# = "Abschluss"
:SET &AKTIVITAET_DATUM# = SYS_DATE()

:PRINT "&AKTIVITAET#datum"
```

Nicht zulässig sind "&AKTIVITAET_DATUM" und "&AKTIVITAET", da letztere Bezeichnung nicht eindeutig ist. Variablennamen und auszugebende Zeichen können direkt aneinander gereiht werden, daher könnte in diesem Beispiel sowohl die Bezeichnung "Abschlussdatum" als auch "040101" gemeint sein.

```
:SET &AKTIVITAET = "Abschluss"
:SET &AKTIVITAETdatum = SYS_DATE()

:PRINT "&AKTIVITAETdatum"
```

Arrays

Es besteht die Möglichkeit Arrays im Zusammenhang mit Script-Variablen zu erstellen. Dadurch können mehrere verschiedene Werte in eine Variable gespeichert werden. Der Zugriff auf die einzelnen Werte erfolgt über den Index. Dieser wird als Nummer in []-Klammern nach dem Variablennamen angegeben. Arrays können nur bei der Variablendeklaration mit :DEFINE erstellt werden. Der dritte Parameter gibt dabei die Anzahl der Werte an (Index-Bereich).

Der Name von Arrays darf im Gegensatz zu normalen Script-Variablen maximal 24 alphanumerische Zeichen lang sein.

```
:DEFINE &ARR#, unsigned, 10
:SET &ARR#[5] = 20
```

Die maximale Größe von Arrays ist 99999. Der Index wird immer als positive Zahl ohne Hochkomma angegeben. Der Zugriff auf das erste Element erfolgt mit dem Index 1.

Beachten Sie, dass bei der Erstellung von Arrays, Speicher für alle Elemente reserviert wird. Daher sollten Arrays nicht mit mehr Elementen angelegt werden, als benötigt werden, um Performanceprobleme zu vermeiden.

Felder von Arrays, die noch nicht initialisiert wurden, besitzen als Standardwert "" (Datentyp: string) oder 0 (bei numerischen Datentypen).

Um mehrere Werte direkt in ein Array zu speichern ist das Script-Sprachmittel :FILL zu verwenden. Dabei wird kein Index angegeben. Dies ist beispielsweise hilfreich, um mehrere Spalten eines Variablen-Objektes auszulesen:

```
:FILL &ARR#[] = GET_VAR("VARA","JOBS")
```

Wertzuweisungen

Wertzuweisungen von Script-Variablen werden mit dem Sprachmittel :SET durchgeführt. Der zugewiesene Wert kann mit oder ohne Hochkommata (Script-Literal) angegeben werden (unabhängig vom Datentyp), oder auch der Rückgabewert einer Script-Funktion sein.

```
:DEFINE &ZAHL#, unsigned
:SET &ZAHL# = 1234
:SET &ZAHL# = "1234"
:SET &ZAHL# = ADD(1,2)
```

Script-Variablen können entweder mit einem bestimmten Datentyp (:DEFINE) oder erst direkt bei der Wertzuweisung erstellt werden.
Variablen, die mit :SET erzeugt werden, besitzen keinen bestimmten Datentyp, können allerdings nur positive Ganzzahlen oder Zeichenketten speichern. Wird einer solchen Variable ein negativer Wert zugewiesen, kommt es zu einem Laufzeitfehler. Wird der Variable eine positive Gleitkommazahl zugewiesen, werden die Kommastellen abgeschnitten.

Bei Script-Variablen, die einen Datentyp aufweisen sind folgende Besonderheiten zu beachten:

Entspricht der zugewiesene Wert nicht dem Datentyp der Script-Variable, so wird versucht diesen umzuwandeln bzw. anzupassen. Ist dies nicht möglich, so kommt es zu einem Laufzeitfehler. Dabei handelt es sich konkret um folgende 2 Fälle:

Fehler-Fall 1: Einer Script-Variable mit dem Datentyp "unsigned" wird eine negative Zahl zugewiesen.
Beispiel:

```
:DEFINE &ZAHL#, unsigned
:SET &ZAHL# = -1
```

Fehler-Fall 2: Einer Script-Variable mit einem Zahlen-Datentyp (unsigned, signed oder float) wird eine Zeichenkette zugewiesen, die keiner Zahl entspricht.
Beispiel:

```
:DEFINE &ZAHL#, signed
:SET &ZAHL# = "abc123"
```

Entspricht die Zeichenkette einer Zahl, so funktioniert die Zuweisung.
Beispiel:

```
:DEFINE &ZAHL#, signed
:SET &ZAHL# = "-123"
```

Wird einer Variable eine Gleitkommazahl zugewiesen, obwohl deren Datentyp dies nicht unterstützt (betrifft: "signed" und "unsigned"), so werden die Nachkommastellen abgeschnitten. Es wird nicht gerundet!
Beispiel: Das Ergebnis der Zuweisung ist in diesem Fall -10.

```
:DEFINE &ZAHL#, signed
:SET &ZAHL# = -10.654
:P &ZAHL#
```

Report:

```
-0000000000000010
```

Zahlen besitzen standardmäßig das 16-stellige Standard-Format. Bei Gleitkommazahlen (Datentyp: float) sind auch 16 Nachkommastellen vorhanden. Stellen, die nicht verwendet werden, werden mit Nullen aufgefüllt. Um führende oder abschließende Nullen zu entfernen, verwenden Sie die Script-Funktion FORMAT.

1.3.4 Script-Literal

Ein Script-Literal besteht aus einer beliebigen Zeichenfolge, die in einfache (') oder doppelte Hochkommata (") eingeschlossen ist.

Beispiel:

```
"Automation Engine"
```

In einem Script-Literal können auch Script-Variablen enthalten sein, die durch ihren Wert ersetzt werden, wenn die Script-Zeile mit dem Script-Literal abgearbeitet wird.

Beispiel:

```
:SET &ZEIT# = SYS_TIME("HH:MM:SS")
:PRINT "Uhrzeit &ZEIT#"
```

Ausgabe:

```
Uhrzeit 10:30:05
```

Automation Engine Name

Eine Sonderform des Script-Literals bildet der AE-Name. Er kann ohne einfache oder doppelte Hochkommata angegeben werden. Zu den AE-Namen gehören:

- Objektnamen,
- Attribute von Objekten oder
- Kurzformen von Objekttypen

Beispiel:

```
:SET &STATUS# = SYS_HOST_ALIVE(WIN01)
```

Beginnt der AE-Name allerdings mit einem numerischen Zeichen, sind stets Hochkommata erforderlich.

⚠ Beachten Sie, dass die Verwendung der Zeichenfolge <![[ ]]> in Script-Literalen zu einem Syntaxfehler führt und das Objekt nicht gespeichert werden kann. Um diesen Fehler zu umgehen gibt es verschiedene Lösungswege:

1. Einfügen eines Kommentars, das die Zeichen enthält.
   Beispiel:
   ```
   !<![[ ]]>
   :PRINT '<![[ ]]>'
   ```

2. Da es nur zu dem Fehler kommt, wenn die Zeichen hintereinander vorkommen, ist es auch möglich die Zeichenkette aus 2 verschiedenen Scriptliteralen zu bilden.
   Beispiel: Verwendung durch Scriptvariablen
   ```
   :SET &VAR# = "]]"
   :PRINT '<![[ &VAR#>'
   ```

   Beispiel: Mit Hilfe des Scriptsprachmittels STR_CAT
   ```
   :SET &VAR# = STR_CAT("<![[ ]]",">")
   :PRINT &VAR#
   ```

⚠ Eine weitere Besonderheit stellt die reservierte Kombination **##***<Zahl>* dar. Wird diese Zeichenfolge in einem Script-Literal verwendet, so wird der Text jener Meldung eingefügt, deren Nummer für *<Zahl>* angegeben wurde. Beachte Sie dies, bei der Verwendung des Script-Literals!

Beispiel:

```
:PRINT "##1800"
```

oder

```
:PRINT ##1800
```

führt zu folgendem Ergebnis im Report:

```
2011-06-15 13:01:51 - U0020408 ENDED_NOT_OK - abgebrochen|
```

1.3.5 Script-Registerkarten

Das AE Script wird in sogenannten Script-Registerkarten niedergeschrieben. Alle ausführbaren Objekte besitzen eine oder mehrere dieser Registerkarten.

Eine Gesamtübersicht über alle Script-Registerkarten ist im Folgenden aufgelistet:

- **Registerkarte Script (alle aktivierbaren Objekte)**

  Der Zeitpunkt der Abarbeitung des Scriptes hängt von der Option "Generieren zur Laufzeit" ab (Registerkarte "Attribute"). Ist diese Einstellung nicht gesetzt, so wird das Script gleich beim Start des Objektes abgearbeitet. Im umgekehrten Fall erfolgt die Abarbeitung des Scriptes erst bei der Durchführung des Objektes.

  ℹ Der Start- und der Durchführungszeitpunkt eines Objektes muss nicht immer übereinstimmen. Dann z.B. wenn eine früheste Startzeit definiert worden ist.

- **Registerkarten Pre- und Post-Script (FileTransfer, Job und RemoteTaskManager)**

  Jobs besitzen zusätzlich noch eine Pre- und eine Post-Script-Registerkarte. Sie werden, wie die Namen schon vermuten lassen, in der folgenden Weise abgearbeitet: Zuerst Pre-Script dann Script. Der Zeitpunkt richtet sich nach der Einstellung "Generierung zur Laufzeit". Nach Beendigung des Jobs wird das Post-Script durchgeführt (siehe auch: Ablauf der Jobdurchführung). Der FileTransfer und RemoteTaskManager besitzen beide nur ein Post-Script, das aber dieselbe Funktion wie beim Job ausübt.

- **Registerkarte !Script (Ereignis)**

  Die Abarbeitung des !Scriptes erfolgt immer dann, wenn das Ereignis eintritt.

Beachten Sie, dass es auch einen eigenen Objekttyp namens "Script" gibt.

Jede Script-Registerkarte kann bis zu 32767 Zeilen umfassen. Dennoch werden standardmäßig nur die ersten 1000 Zeilen berücksichtigt, um Endlosschleifen bei der Generierung zu vermeiden und damit den AE Server zu schützen. Übersteigt das Script 1000 Zeilen wird die Generierung abgebrochen. Diese Zeilenbegrenzung können Sie mit dem Attribut MAX_JCL_LINES, das mit der Script-Anweisung :PUT_ATT gesetzt werden kann, ändern.

1.3.6 Scripteditor

Jede Script-Registerkarte ist mit einem Scripteditor ausgestattet. Er stellt eine Vielzahl von Funktionalitäten bereit, die Ihnen die Erstellung von Scripts komfortabel machen.

Hervorheben von Syntaxkomponenten

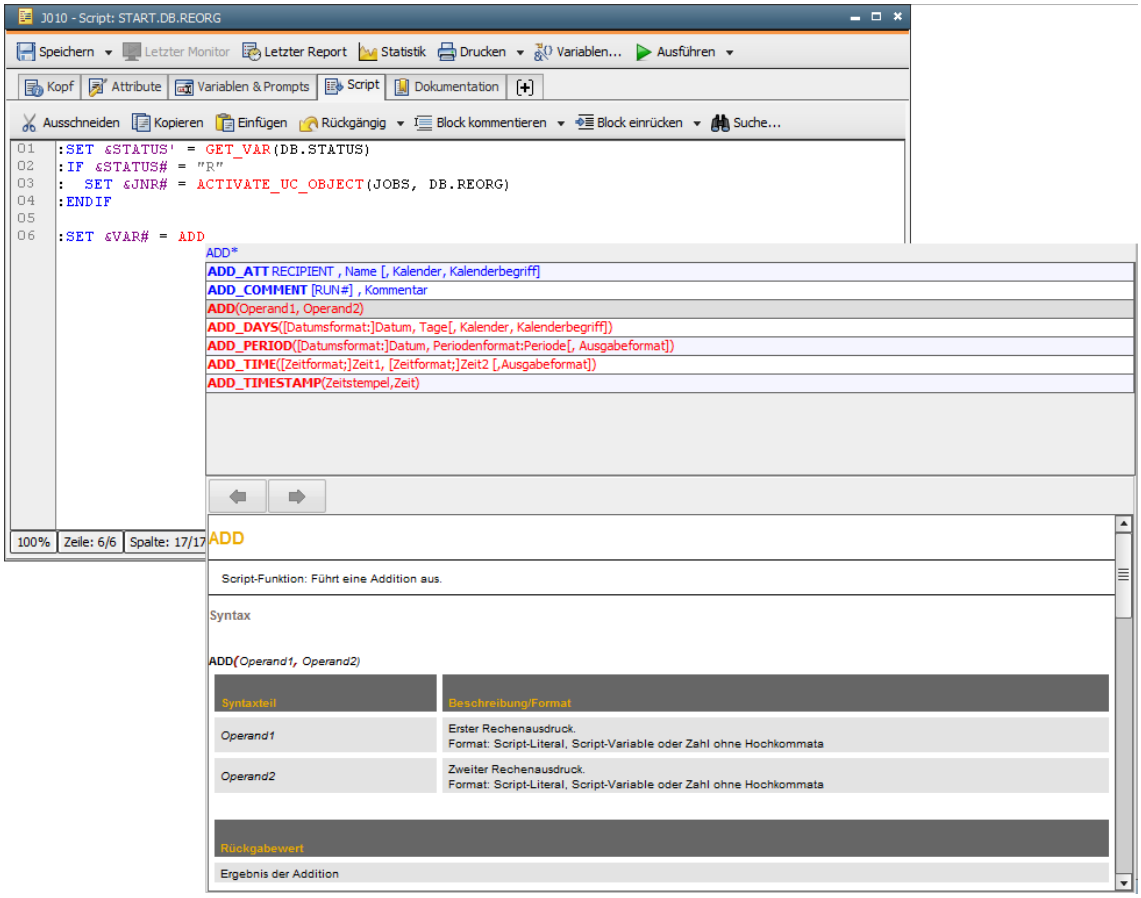
Die einzelnen Teile des Scripts sind zur besseren Unterscheidung farblich hervorgehoben:

- grün für Kommentarzeilen
- blau für Script-Anweisungen
- rot für Script-Funktionen
- braun für JCL-Zeilen
- grau für Zeichenfolgen
- lila für Script-Variablen

Der Scripteditor hebt auch alle Vorkommen einer Script-Variablen oder eines Script-Sprachmittels hervor, wenn Sie den Mauszeiger im Namen positionieren.

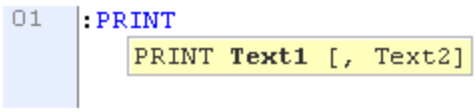
Autovervollständigung

Sie haben die Möglichkeit die Autovervollständigung mit der Tastenkombination STRG + Leertaste aufzurufen. Sie komplettiert den Namen von Script-Funktionen und Script-Anweisungen. Kommen mehrere in Frage, wird eine Liste angezeigt aus der Sie das gewünschte Script-Sprachmittel auswählen können. Die ESC-Taste schließt die Liste der Autovervollständigung.



Der Scripteditor zeigt zusätzlich zur Liste auch die Seite der AE-Dokumentation an, die eine Beschreibung zum markierten Script-Sprachmittel enthält, wenn Sie die entsprechende Option in den Einstellungen des UserInterfaces aktivieren.

Nachdem Sie den Namen des Script-Sprachmittels ausgeschrieben bzw. direkt aus der Liste ausgewählt haben, werden die Parameter eingeblendet, wenn Sie die entsprechende Option in den Einstellungen des UserInterfaces aktivieren. Der Parameter, der an der Stelle einzugeben ist an dem sich der Mauszeiger befindet, wird fett hervorgehoben.



Include-Objekte

Wenn Sie Include-Objekte in Ihr Script einbinden, können Sie sich den Inhalt anzeigen lassen. Klicken Sie hierzu im linken Rand der Registerkarte auf das Pluszeichen. Die eingeblendeten Scriptzeilen des Include-Objektes sind editierbar. Speichern Sie das Script, so werden auch die Änderungen im Include-Objekt gespeichert.

Beachten Sie, dass die Änderung alle Objekte beeinflusst, welche das Include-Objekt verwenden!

Benutzer, die kein Lese- oder Schreibrecht auf das Include-Objekt besitzen, können den Inhalt nicht anzeigen lassen bzw. ändern. In letzterem Fall werden die Scriptzeilen des Include-Objektes hellgrau angezeigt und sind nicht editierbar.

Beachten Sie, dass die Suche im Script den Inhalt von Include-Objekten durchsucht, wenn sie mit Hilfe des Pluszeichens aufgeklappt worden sind. Beim Exportieren des Scripts in eine Textdatei und beim Kopieren von Scriptszeilen, die Include-Objekte enthalten, werden auch nur die Inhalte von aufgeklappten Include-Objekten berücksichtigt.

Wenn Sie ein Include-Objekt mehrfach in einem Script einbinden, so kann es nur an der Stelle geändert werden an der Sie es zuerst aufklappen. Der Scripteditor zeigt die anderen Vorkommen hellgrau an.

Zeilenpräfix und -einrückungen

Der Scripteditor unterstützt Sie bei der schnellen und übersichtlichen Scripterstellung indem er das Präfix der aktuellen Zeile in die Folgezeile übernimmt. Außerdem kann er Scriptzeilen in Konstrukten wie beispielsweise :IF oder :WHILE einrücken und sie mit der erforderlichen Abschlußanweisung abschließen. (z.B: :ENDIF oder :ENDWHILE). Jede dieser Funktionalitäten ist standardmäßig in den Einstellungen des UserInterfaces aktiviert.

Aufruf der AE-Dokumentation

Markieren Sie den Namen einer Script-Funktion oder einer Script-Anweisung und drücken Sie die F1-Taste. Die AE-Dokumentation öffnet sich auf der Seite, die eine Beschreibung zum Script-Sprachmittel enthält.

Es genügt auch den Mauszeiger innerhalb des Namens der Script-Funktion bzw. Script-Anweisung zu platzieren und dann F1 zu drücken.

Kontextmenü-Befehle

Neben den Standardbefehlen wie beispielsweise Rückgängig, Kopieren oder Ausschneiden haben Sie auch Zugriff auf folgende Funktionalitäten:

- Die Suche findet nicht nur gesuchte Zeichenfolgen, sondern bietet auch die Möglichkeit jene durch andere Bezeichnungen zu ersetzen.
- Mehrere Zeilen können gleichzeitig kommentiert und entkommentiert werden.
- Nutzen Sie des Weiteren die Möglichkeit Scripts in Dateien abzulegen bzw. zu importieren.

Befehl	Beschreibung
Ausschneiden Kopieren Einfügen	Befehlen zum Bearbeiten des Textes
In Großbuchstaben In Kleinbuchstaben	Ändert die Schreibweise des markierten Textes
Rückgängig	Macht die letzte Aktion rückgängig
Wiederholen	Nimmt die letzte rückgängig gemachte Aktion zurück

Block kommentieren	Fügt einer oder mehreren Zeilen am Beginn ein Rufzeichen ein, damit sie als Kommentar gekennzeichnet ist.
Block entkommentieren	Entfernt einer oder mehreren Zeilen am Beginn das Rufzeichen, damit sie nicht mehr als Kommentar gekennzeichnet ist.
Block einrücken Block ausrücken	Rückt die Zeile bzw. den markierten Textblock ein bzw. aus. Rufzeichen mit denen Kommentarzeilen und Doppelpunkte mit denen AE Script-Zeilen beginnen, bleiben dabei an der ersten Stelle der Zeile stehen. Der Rest der Zeile wird ein- bzw. ausgerückt.

Alles Reformatieren	Formatiert alle Zeilen der Registerkarte im AE Script Stil.
	Beispiel:
	Die Zeile
	`:  PRINT "Beginn der Verarbeitung"`
	würde wie folgt formatiert werden:
	`:PRINT "Beginn der Verarbeitung"`

Von Datei importieren	Importiert den Inhalt einer Textdatei an die Stelle an der der Mauszeiger positioniert ist
In Datei exportieren	Exportiert den gesamten Inhalt der Registerkarte in eine Datei
Alles selektieren	Markiert den gesamten Inhalt der Registerkarte
Suche	Öffnet den Suchdialog für Textstellen

1.3.7 Starthilfe

Das Kapitel Beispielsammlung beinhaltet Script-Beispiele, deren Anwendung im Zusammenhang mit Objekten erläutert wird. Es bietet für Laien und Experten gleichermaßen einen Einblick in die Script-Sprache von AE.

- Übersicht - Beispielsammlung

Sämtliche Script-Sprachmittel sind in der AE-Dokumentation sowohl alphabetisch als auch funktionell gegliedert. Nutzen Sie die folgenden beiden Übersichten, um rasch jenes Script-Sprachmittel zu finden, das Sie benötigen.

- Übersicht - alphabetische Auflistung
- Übersicht - funktionelle Gliederung

In sämtlichen Script-Registerkarten kann über die F1-Taste die AE-Dokumentation aufgerufen werden. Sie öffnet jene Seite, die eine Beschreibung zum Script-Sprachmittel enthält. Markieren Sie dazu den Namen des Script-Sprachmittels (z.B. PRINT) oder positionieren Sie den Mauscursor in dessen Namen. Somit haben Sie immer einen schnellen Zugriff zur Syntaxbeschreibung.

1.4 Weiterführende Themen

1.4.1 Script-Bausteine benutzen

Viele Objekte in der AE verwenden in ihrem Script gleiche Verarbeitungsschritte. Um diese nicht in jedem einzelnen Script schreiben und pflegen zu müssen, ist es empfehlenswert sie zentral in Include-Objekte abzulegen. Daraus entstehen sogenannte Script-Bausteine, die einfach zu warten sind und die Erstellung von Scripts beschleunigen.

Der Ablauf der Nutzung eines Include-Objekt sieht wie folgt aus:

1. Legen Sie einen Block von Scriptzeilen, die oft zur Anwendung kommen, in einem Include-Objekt ab.
2. Rufen Sie an der Stelle im Script, in der die Scriptzeilen zur Laufzeit eingefügt werden sollen, die Script-Anweisung :INCLUDE mit dem Namen des Include-Objektes auf.
3. Das Script-Sprachmittel enthält auch einen Parameter, mit der Sie die Scriptzeilen speziell auf das Objekt abstimmen können. Dieser erlaubt nämlich die Ersetzung einer bestimmten Zeichenfolge. Die Zuweisung ist nur für die laufende Generierung gültig und ändert das Include-Objekt selbst nicht.

⚠ Die Scriptzeilen des Include-Objektes werden genau an die Scriptstelle des Objektes kopiert an der Sie :INCLUDE aufrufen. Die Script-Funktionen beziehen sich daher auf das jeweilige Objekt und nicht auf das Include-Objekt selbst. Beispielsweise liefert die Script-Funktion SYS_ACT_ME_NAME den Namen des Objektes und nicht den Namen des Include-Objektes.

⚠ Im Scripteditor kann der Inhalt von eingebundenen Include-Objekte angezeigt und bearbeitet werden. Dies ist durch Aufklappen des Includes über das plus-Symbol am linken Rand möglich. Bearbeiten Sie das Script eines aufgeklappten Includes und speichern das Objekt, so werden die Änderungen auch für das Include-Objekt übernommen! Dies beeinflusst daher auch alle Objekte, welche dieses Include-Objekt verwenden.
Über die Rechtevergabe der Benutzer (Schreibrecht für Include-Objekte), können Sie ungewünschte Änderungen an Include-Objekten unterbinden.

1.4.2 Fehlerbehandlung

Bei der Erstellung eines AE Scriptes sollten Sie unbedingt daran denken, dass Fehler auftreten können. Aus diesem Grund empfielt es sich Vorsichtsmaßnahmen zu treffen.

Erstellung eines Scripts

Die allererste Fehlerquelle liegt bereits in der Verfassung eines AE Scriptes. Deshalb wird beim Speichern des Objektes die Syntax der verwendeten Script-Funktionen und -Anweisungen geprüft. Falls Sie beispielsweise zuviele bzw. zuwenige Parameter angeben, erscheint eine Fehlermeldung, die Sie auf diesen Umstand hinweist. Sie enthält auch die Zeilennummer in der der Fehler aufgetreten ist.

⚠ Die Prüfung des Scriptes betrifft die Syntax. Für die richtigen Werte hingegen müssen Sie selbst sorgen (z.B. korrekte Objektnamen).

Nutzung von Rückgabewerten

Rückgabewerte von Funktionen geben über das Ergebnis der Script-Funktion Aufschluss. Machen Sie sich das zunutze, indem Sie diese auswerten. Welche Rückgabewerte eine Funktion liefern kann, ist bei der Beschreibung zum Script-Sprachmittel vermerkt.

Verwendung von :ON_ERROR

Tritt ein Fehler bei der Durchführung des Scriptes auf, so wird eine Fehlermeldung angezeigt und das Script abgebrochen. Einige Script-Sprachmittel sind davon jedoch ausgenommen. Mit :ON_ERROR ist in solchen Fällen steuerbar, ob ein Scriptabbruch erfolgen soll. Wenn nicht, so können Sie mit speziellen Script-Funktionen wie SYS_LAST_ERR_NR und SYS_LAST_ERR_INS auf die Fehlermeldung zugreifen.

1.4.3 Verwendung von Variable-Objekten

Werte sind nicht nur in Script-Variablen, sondern auch in statischen Variable-Objekten speicherbar. Bei einer großen Anzahl von Werten ist diese zweite Methode überaus komfortabel. Das AE Script stellt eine Reihe von Script-Sprachmitteln bereit, die das Hinterlegen, Auslesen und Löschen von Werten in Variable-Objekten ermöglichen.

Der Wert wird immer zusammen mit einem Key abgelegt, über den ein Zugriff auf jenen stattfinden kann.

Die Handhabung von Variable-Objekten kann entweder manuell im Explorer des UserInterfaces erfolgen oder mit Hilfe folgender Script-Sprachmitteln:

- CREATE_OBJECT - Legt ein neues Variable-Objekt an.
- MODIFY_OBJECT - Ändert die Attribute eines Variable-Objekt.
- REMOVE_OBJECT - Löscht ein Variable-Objekt.

Befüllen mit Werten

Das Speichern von Werten geschieht durch die Script-Anweisung :PUT_VAR. Der Wert wird hinzugefügt, wenn der Key noch nicht existiert. Ansonsten wird der bestehende Wert ersetzt.

Auslesen von Werten

Mit der Script-Funktion PREP_PROCESS_VAR können Sie sowohl einen einzelnen als auch mehrere oder alle Werte eines Variable-Objektes ermitteln.

Löschen von Werten

Das Entfernen von Werten aus einem Variable-Objekt bewirkt die Script-Anweisung :DELETE_VAR.

1.4.4 Objekt-Attribute verändern

Jedes Objekt besitzt eine große Anzahl an Attributen, welche die Verarbeitung beeinflussen. Dazu gehören z.B. die Priorität oder der Starttyp. Oftmals treten Situationen auf, die eine Änderung eines Attributes notwendig werden lassen. So könnte es beispielsweise bei einem FileTransfer nötig sein den Zielhost zu ändern, weil eine bestimmte Kalenderbedingung zutrifft oder der Host nicht aktiv ist.

Das AE Script beinhaltet daher die Script-Sprachmittel :PUT_ATT und GET_ATT, die die Attribute eines Objektes setzen und auslesen können.

```
:PUT_ATT GROUP = "MAWI.GRUPPE"
```

```
:SET &START# = GET_ATT(GROUP)
```

Im Benutzerhandbuch finden Sie eine Liste der Attribute aller Objekte mit ihren möglichen Werten.

Für Benachrichtigungs-Objekte gibt es noch zusätzliche Script-Sprachmittel:

- :ADD_ATT - Fügt einem Benachrichtigungs-Objekt Empfänger zur Laufzeit hinzu.
- :REMOVE_ATT - Entfernt in einem Benachrichtigungs-Objekt Empfänger zur Laufzeit.
- :PUT_ATT_APPEND - Erweitert den Meldungstext einer Benachrichtigung zur Laufzeit.
- GET_ATT_SUBSTR - Liefert einen Teil des Meldungstextes einer Benachrichtigung.

1.4.5 Rückgabewerte von Funktionen

Script-Funktionen liefern im Gegensatz zu Script-Anweisungen Rückgabewerte, die in Script-Variablen gespeichert werden. Rückgabewerte können einzelne Zeichen, ganze Zeichenfolgen oder Zahlen sein. Beachten Sie, dass letztere immer 16-stellig mit führenden Nullen dargestellt werden.

Beispiel:

```
:SET &SUMME# = ADD(2,2)
 :PRINT &SUMME#
```

Das mit PRINT angezeigte Ergebnis sieht wie folgt aus:

```
0000000000000004
```

Wenn Sie Zahlen ausgeben, z.B. in den Report, können Sie die führenden Nullen mit der Script-Funktion FORMAT entfernen. Für die Weiterverarbeitung der Zahl (z.B. Rechenoperationen) haben die Nullen jedoch keine Auswirkung.

Der Rückgabewert kann auch eine Fehlernummer enthalten. Welche das ist und was sie bedeutet, ist bei der Beschreibung zum Script-Sprachmittel vermerkt. Es gibt bestimmte Script-Sprachmittel, die eine weitere Behandlung der Fehlernummer ermöglichen.

Einige Script-Sprachmittel müssen gemeinsam verwendet werden. Die Script-Funktion PREP_PROCESS_FILE beispielsweise erzeugt eine Datensequenz aus dem Inhalt einer Textdatei. Der Rückgabewert ist ein sogenannter Verweis (Handle) mit dem sich alleine nichts weiter tun lässt. Die einzelnen Zeilen der Textdatei können aber mit GET_PROCESS_LINE ausgelesen werden. Diese Script-Funktion benötigt jedoch dazu den mit PREP_PROCESS_FILE ermittelten Verweis.

1.4.6 Rechnen

Dieses Dokument erklärt die Rechenoperationen die mit Hilfe des AE Scripts durchgeführt werden können.

Scriptsprachmittel

Im AE Script können Sie mit Hilfe von Script-Sprachmittel die 4 Grundrechnungsarten verwenden:

- Addition
- Subtraktion
- Multiplikation
- Division

Zusätzlich steht die Script-Funktion MOD zu Ihrer Verfügung, die den Rest einer Division ermittelt.

Mit der Script-Funktion RANDOM können Sie bei Bedarf Zufallszahlen erzeugen.

Das Resultat der Rechenoperation wird in einer Script-Variablen gespeichert.

Beispiel:

```
:SET  &DIFFERENZ# = SUB(100,50)
```

Bei Rechenoperationen werden auch negative Zahlen (Datentyp: "signed") und Kommazahlen (Datentyp: "float") unterstützt. Dazu muss die Zielvariable, die das Ergebnis der Rechnung erhält, den geeigneten Datentyp aufweisen. Der Datentyp "string" ist für Rechenoperationen nicht erlaubt!

Nachkommastellen der Ergebnisse werden abgeschnittten, wenn der Datentyp der Zielvariable keine Kommazahlen unterstützt.

Wenn eine Rechenoperation ein negatives Ergebnis liefert und der Datentyp der Zielvariable keine Vorzeichen unterstützt, kommt es zu einem Laufzeitfehler!

Es wird daher empfohlen, als Datentyp der Zielvariable den höchsten verwendeten Datentyp der Rechnung zu verwenden (siehe Tabelle). Das Ergebnis der Rechnung darf den Wert "9 999 999 999 999 999" nicht übersteigen.

Datentypen der Operanden	Erlaubte Datentypen der Zielvariable
unsigned, unsigned	unsigned, signed, float
unsigned, signed	signed, float
unsigned, float	float
signed, signed	signed, float
signed, float	float
float, float	float

Der höchste Datentyp ist "float" (Negative und positive Kommazahlen). Danach kommt "signed" (Negative und positive Ganzzahlen). Der Datentyp "unsigned" unterstützt nur positive Ganzzahlen und ist somit am niederwertigsten.

Beispiel: Der Datentyp der Ergebnis-Variable "&SUM#" muss mindestens "signed" sein, wenn als Operand eine negative Zahl angegeben wurde. Es ist jedoch auch der Datentyp"float" möglich.

```
:DEFINE  &UNSIGNED#,unsigned
:DEFINE  &SIGNED#,signed
:DEFINE  &SUM#,signed

:SET &UNSIGNED# = 12
:SET &SIGNED# = -5

:SET  &SUM# = ADD(&SIGNED#,&UNSIGNED#)
```

Diese Zeile würde zu einem Scriptfehler resultieren, wenn die Variable &signed# einen negativen Wert enthält. Ist der Variablenwert jedoch positiv, funktioniert die Operation.

```
:SET  &UNSIGNED# = ADD(&SIGNED#,&UNSIGNED#)
```

⚠ Beachten Sie, dass es bei Rechenoperationen mit Gleitkommazahlen zu einem ungenauen Ergebnis kommen kann!

Auflösung eines arithmetischen Terms

in AE ist es auch möglich mit Hilfe des Script-Sprachmittels :SET einen arithmetischen Term aufzulösen und das Ergebnis in eine Variable zu speichern. Somit wird das Script um ein Vielfaches zu verkürzt und vereinfacht, da nicht für jede Rechenoperation ein eigenes Scriptsprachmittel notwendig ist.

⚠ Weist die Zielvariable den Datentyp "unsigned" auf und das Ergebnis der Rechnung ergibt eine negative Zahl, führt dies zu einem Laufzeitfehler. Unterstützt der Datentyp der Zielvariable keine Nachkommastellen, so werden diese gegebenenfalls abgeschnitten.

Die Rechnung kann neben den Grundrechnungsarten auch Klammern und Vorzeichen beinhalten.

- Addition
- Subtraktion
- Multiplikation
- Division
- Klammern "()"
- Vorzeichen

Eine Zeile die Rechenoperatoren (+,-,*,/) enthält, wird grundsätzlicher als Term behandelt. Achten Sie darauf, den Term nicht unter einzelne oder doppelte Hochkommata zu setzen. Andernfalls wird der Term als string interpretiert.

⚠ Die Verwendung von Script-Funktionen innerhalb arithmetischer Ausdrücke ist nicht erlaubt.

Achten Sie auf die Gewichtung der Operatoren (Punktrechnung vor Strichrechnung)

Beispiel:

```
 :DEFINE  &UNSIGNED#,unsigned
:DEFINE  &FLOAT#,float
:DEFINE  &RES#,float

:SET &UNSIGNED# = 12
:SET &FLOAT# = -0.50

:SET &RES# = &FLOAT#*3 + (-&UNSIGNED#) - 3
```

Siehe auch:

Script-Sprachmittel	Beschreibung
ADD	Führt eine Addition aus.
SUB	Führt eine Subtraktion aus.
MULT	Führt eine Multiplikation aus.
DIV	Führt eine Division aus.
MOD	Liefert den Restwert einer Division.
RANDOM	Generiert Zufallszahlen.
GET_BIT	Prüft, ob in einem Bit-Feld ein bestimmtes Bit gesetzt ist.

Script-Sprachmittel - Rechnen

Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

1.4.7 Zeichenketten

Das AE Script enthält auch eine Reihe von Sprachmitteln mit denen Sie Zeichenketten, sogenannte Strings, bearbeiten können. Die wichtigsten möchten wir Ihnen hier vorstellen. Eine Liste von allen String-Funktionen finden Sie in der funktionellen Übersicht.

Länge der Zeichenkette ermitteln

Beschreibung	Beispiel	Ergebnis
STR_LENGTH - Gibt die Anzahl der Zeichen zurück.	`:SET &LAENGE# = STR_LENGTH ("UC4")`	3

Zeichen vergleichen

Beschreibung	Beispiel	Ergebnis
STR_MATCH - Vergleicht zwei Zeichenketten miteinander.	`:SET &VERGLEICH# = STR_MATCH ("UC4", "System")`	N

Zeichen verbinden

Beschreibung	Beispiel	Ergebnis
STR_CAT - Verbindet zwei Zeichenketten.	`:SET &STRING# = STR_CAT("AE", "-System")`	Automation Engine System

Zeichen suchen

Beschreibung	Beispiel	Ergebnis

STR_FIND - Gibt die Stelle zurück, an der sich die gesuchte Zeichenkette befindet.	`:SET &STELLE# = STR_FIND ("Automation Engine System", "System")`	5

Zeichen ersetzen

Beschreibung	Beispiel	Ergebnis
STR_SUBSTITUTE - Ersetzt einen Teil einer Zeichenkette.	`:SET &STRING# = STR_SUBSTITUTE ("UC4-Umgebung", "Umgebung", "System")`	Automation Engine System

Zeichen ausschneiden

Beschreibung	Beispiel	Ergebnis
SUBSTR - Gibt einen Teil der Zeichenkette zurück	`:SET &TEILSTRING# = SUBSTR ("Automation Engine System", 5)`	System

Zeilenumbruch

Beschreibung	Beispiel	Ergebnis
UC_CRLF - Liefert einen Zeilenumbruch.	`:SET &NL# = UC_CRLF()` `:SET &STRING# = "AE &NL#-System"`	Automation Engine System

1.4.8 Datums-, Zeit- und Periodenformate

In einigen Script-Anweisungen und Script-Funktionen können Sie Formate für Datums-, Zeit- oder Periodenformate angeben. Welche hierbei erlaubt sind, ist in den folgenden Tabellen aufgelistet.

Datumsformate

[Datumsformate] [Zeitformate] [Periodenformate]

In den Datumsformaten werden die Abkürzungen "J" oder "Y" für das Jahr, "M" für den Monat sowie "T" oder "D" für den Tag verwendet.

Datumsformat	Beispiel/Vorschau
JJMMTT oder YYMMDD	001231
JJ.MM.TT oder YY.MM.DD	00.12.31
JJ-MM-TT oder YY-MM-DD	00-12-31
JJJJMMTT oder YYYYMMDD	20001231

JJJJ.MM.TT oder YYYY.MM.DD	2000.12.31
JJJJ-MM-TT oder YYYY-MM-DD	2000-12-31
TTMMJJ oder DDMMYY	311200
TT.MM.JJ oder DD.MM.YY	31.12.00
TT-MM-JJ oder DD-MM-YY	31-12-00
TTMMJJJJ oder DDMMYYYY	31122000
TT.MM.JJJJ oder DD.MM.YYYY	31.12.2000
TT-MM-JJJJ oder DD-MM-YYYY	31-12-2000
MMDDYY	123100
MMDDYYYY	12312000
MM/DD/YY	12/31/00
MM/DD/YYYY	12/31/2000

Datumsformate können auch als Teilbegriffe angegeben werden. Zwei besondere Teilbegriffe sind "WW" und "LLL". "WW" formatiert auf den Wochentag als 2-stelliges Kürzel. "LLL" entspricht dem laufenden Tag im Jahr.

Die Verwendung von Teilbegriffen der Datumsformate ist bei Script-Funktionen nicht erlaubt, die mit einem Datum rechnen. Ausnahme ist das Ausgabeformat eines ermittelten Datums.

Datumsformat (Teilbegriffe)	Beispiel/Vorschau
JJ oder YY	00
JJJJ oder YYYY	2000
MM	12
TT oder DD	31
WW	FR
LLL	365

Zeitformate

[Datumsformate] [Zeitformate] [Periodenformate]

In den Zeitformaten wird die Abkürzung "H" für die Stunde, "M" für die Minute und "S" für die Sekunde verwendet.

Zeitformat	Beispiel/Vorschau
HHMMSS	235959
HH:MM:SS	23:59:59
HHMM	2359
HH:MM	23:59
MMSS	5959
MM:SS	59:59

Zeitformat (Teilbegriffe)	Beispiel/Vorschau
HH	23
MM	59
SS	59

Periodenformate

[Datumsformate] [Zeitformate] [Periodenformate]

In den Periodenformaten wird die Abkürzung "J" oder "Y" für das Jahr, "Q" für das Quartal, "M" für den Monat und "W" für die Kalenderwoche verwendet.

Die folgende Periodenformate können bei den Script-Sprachmittel FIRST_OF_PERIOD, LAST_OF_PERIOD, ADD_PERIOD, SUB_PERIOD eingesetzt werden.

Periodenformat	Beispiel/Vorschau
JJ oder YY	00
JJJJ oder YYYY	2000
Q	3
MM	06
WW	53
WS *	53

* Das Periodenformat "WS" kann nur in den Script-Funktionen zur Ermittlung des ersten oder letzten Tages einer Periode genutzt werden. Für dieses Periodenformat gilt Sonntag als erster Tag der Woche. Im Gegensatz dazu geht "WW" vom Montag als ersten Tag der Woche aus.

1.4.9 Benachrichtigungen verschicken

Benachrichtigungen sind ein gutes Mittel, um auf Verarbeitungsergebnisse aufmerksam zu machen. Diese können mit Scriptsprachmittel auf 4 verschiedene Arten versendet werden.

Art	Beschreibung
Benachrichtigungs-Objekt	Die gängigste Methode ist die Aktivierung eines Benachrichtigungs-Objektes. Mehrere AE-Benutzer können auf einmal verständigt werden. Verwenden Sie die Script-Funktion ACTIVATE_UC_OBJECT, um ein Benachrichtigungs-Objekt zu aktivieren.
SEND_MAIL	Es ist auch möglich mittels der Script-Funktion ein E-Mail zu senden, das an beliebige Empfänger geschickt werden kann.
:SEND_MSG	Für kurze Informationsnachrichten eignet sich diese Script-Anweisung. Sie erscheinen im Meldungsfenster eines AE-Benutzers.

:SEND_SNMP_TRAP	Wenn Sie die Integrationsmöglichkeiten von AE in Frameworks nutzen, ist diese Script-Anweisung hilfreich. Sie ermöglicht das Versenden von benutzerdefinierten Traps.

1.4.10 Scriptverarbeitung

Die Automation Engine arbeitet ein Script Zeile für Zeile ab. In regelmäßigen Abständen wird das Ergebnis der durchgeführten Script-Sprachmitteln in die Datenbank geschrieben (z.B: der mit :PUT_VAR gesetzte Wert für ein Variablen-Objekt).

Dieser Vorgang wird auch als Commit bezeichnet. Andere Scripts haben erst dann Zugriff die neuen bzw. geänderten Werte, wenn sie in der Datenbank eingetragen worden sind.

Die Automation Engine führt automatisch alle fünf Sekunden ein Commit durch. Zusätzlich wird es auch bei folgenden Script-Sprachmitteln vorgenommen:

:BEGINREAD... :ENDREAD	FORECAST_OBJECT	PREP_PROCESS_FILE
:READ	FORECAST_TASK	PREP_PROCESS_FILENAME
:SET_UC_SETTING	GET_FILESYSTEM	PREP_PROCESS_REPORT
:WAIT	GET_UC_SETTING	REMOVE_OBJECT
ACTIVATE_UC_OBJECT	IMPORT	RESTART_UC_OBJECT
AUTOFORECAST	MODIFY_OBJECT	SEND_MAIL
CANCEL_UC_OBJECT	MODIFY_UC_OBJECT	SYS_SERVER_ALIVE
CREATE_OBJECT	MOVE_OBJECT	TOGGLE_SYSTEM_STATUS
EXPORT	PREP_PROCESS	

Mit der Anweisung :WAIT 0 erzwingen Sie ein Commit.

1.5 Nicht länger unterstützte Script-Sprachmittel

In den AE-Versionen wurden Änderungen an einigen Script-Sprachmitteln vorgenommen.

Die folgende Liste zeigt um welche es sich handelt und wie sich die Anpassungen auswirken:

- :REPLACE_JP_STRUCTURE

  Diese Script-Anweisung wurde mit der Version 8.00A auf :REPLACE_STRUCTURE umbenannt. Die alte Schreibweise wird weiterhin unterstützt

- :SET_UC_SETTING

  Die Einstellung MAX_PARALLEL wurde in einer früheren Version der Automation Engineauf WORKLOAD_MAX umbenannt. Die alte Schreibweise wird weiterhin unterstützt.

- :XML_CLOSE_DOCU

  Diese Script-Anweisung wurde in einer früheren Version der Automation Engineauf :XML_CLOSE umbenannt. Die alte Schreibweise wird weiterhin unterstützt.

- GET_UC_SETTING

  Die Einstellung MAX_PARALLEL wurde in einer früheren Version der Automation Engineauf WORKLOAD_MAX_JOB umbenannt. Beim Update der Datenbank werden die Scripts automatisch angepasst.

- PREP_PROCESS_HOSTGROUP

  Diese Script-Funktion wurde mit der Version 8.00A auf PREP_PROCESS_ AGENTGROUP umbenannt. Die alte Schreibweise wird weiterhin unterstützt.

- XML_OPEN_DOCU

  Diese Script-Funktion wurde in einer früheren Version der Automation Engineauf XML_ OPEN umbenannt. Die alte Schreibweise wird weiterhin unterstützt..

- CINT und CSTR

  Die Script-Funktionen werden nicht für negative Zahlen und Gleitkommazahlen unterstützt. Die Funktion der Umwandlung in alle Datentypen übernimmt seit der Version 9.00A das Script-Sprachmittel CONVERT.

- SYS_ACT_JPNAME und SYS_ACT_JPNR

  Diese Scriptsprachmittel besitzen die selbe Funktion wie SYS_ACT_PARENT_NAME und SYS_ACT_PARENT_NR. Die Verwendung ist jedoch weiterhin möglich.

- SYS_ACT_JOBNAME und SYS_ACT_JOBNR

  Gleiche Funktion wie SYS_ACT_ME_NAME und SYS_ACT_ME_NR. Die Script-Funktionen werden jedoch weiterhin unterstützt.

2 Script-Sprachmittel - alphabetische Auflistung

Hier finden Sie eine alphabetische Auflistung sämtlicher Scriptsprachmittel und deren Anwendung.

Script-Anweisungen

[Script-Anweisungen] [Script-Funktionen]

Script-Anweisung	Beschreibung
:ADD_ATT	Fügt einem Benachrichtigungs-Objekt Empfänger zur Laufzeit hinzu
:ADD_COMMENT	Fügt einer Aufgabe einen Kommentar hinzu
:ATTACH_SYNC	Weist einer Aufgabe ein Sync-Objekt zu
:BEGINREAD... :ENDREAD	Beginn und Ende eines Dialogfeldes für Benutzerabfragen
:CLEAR	Setzt ein Script-Array wieder auf seine Initialwerte zurück.
:CLOSE_PROCESS	Verwirft eine Datensequenz innerhalb eines Scripts
:CONST	Erstellt eine Script-Variable als Konstante mit einem bestimmten Wert.
:DATA	Ausdrückliche Deklaration einer DATA-Zeile im Script
:DEFINE	Erstellung einer Script-Variable mit einem bestimmten Datentyp.
:DELETE_VAR	Löscht einen oder alle Werte aus einem statischen Variablen-Objekt
:DISCONNECT	Trennt eine Verbindung zum AE-System
:EXIT	Beendet die Prozessierung des Scripts mit Rückgabewert
:EXT_REPORT_OFF	Schaltet das Protokollieren des Scripts einer Aufgabe aus
:EXT_REPORT_ON	Schaltet das Protokollieren des Scripts einer Aufgabe ein
:FILL	Speichert mehrere Werte in ein Script-Array.
:GENERATE	Steuert die Behandlung von Script-Zeilen bei der Prozessierung eines Scripts
:IF... :ELSE... :ENDIF	Verzweigung in Abhängigkeit von Bedingungen
:INCLUDE	Bindet ein Include-Objekt in das aktuelle Script ein
:INC_SCRIPT	Bindet ein Script in ein anderes Script des gleichen Objektes ein
:JCL_CONCAT_CHAR	Bildet JCL-Zeilen bis zu einer maximalen Länge von 2 KB
:JCL_SUBSTITUTE	Ersetzt eine Zeichenkette in der JCL durch eine andere Zeichenkette
:MODIFY_STATE	Ändert Rückgabewert oder Statustext eines Jobs nach dessen Ende
:ON_ERROR	Legt die Reaktion auf bestimmte Fehler bzw. Meldungen von Script-Sprachmitteln fest
:PRINT	Gibt Text in einem Dialogfeld für Benutzerabfragen oder im Aktivierungsprotokoll eines Objektes aus

:PROCESS... :TERM_PROCESS... :ENDPROCESS	Schleife zur zeilenweisen Bearbeitung einer Datensequenz, beispielsweise sequentielle Datei oder Kommandoergebnis
:PSET	Weist einer Objektvariablen einen Wert zu.
:PUBLISH	Script-Variablen und Arrays als Objektvariablen definieren.
:PUT_ATT	Setzt oder ändert Attribute von Objekten
:PUT_ATT_APPEND	Erweitert den Meldungstext einer Benachrichtigung zur Laufzeit
:PUT_PROMPT_ BUFFER	Stellt Namen und Inhalt einer Script-Variablen in einen Zwischenspeicher
:PUT_READ_ BUFFER	Stellt Namen und Inhalt einer Script-Variablen in einen Zwischenspeicher
:PUT_VAR	Speichert Werte in ein statisches Variablen-Objekt
:PUT_VAR_COL	Speichert einen Wert in eine bestimmte Spalte eines statischen Variablen-Objektes
:READ	Fragt Angaben vom Benutzer im Dialog ab
:REGISTER_ OUTPUTFILE	Registriert eine Datei als externen Job-Output
:REMOVE_ATT	Entfernt in einem Benachrichtigungs-Objekt Empfänger zur Laufzeit
:REPLACE_ STRUCTURE	Ersetzt die Struktur eines Workflows bei dessen Aktivierung durch die Struktur eines anderen Workflows.
:RESTART	Setzt Restartpunkte in einem Objekt
:RSET	Weist einer Script-Variablen einen Wert zu und speichert diesen im Aktivierungsreport
:SEND_MSG	Schickt Nachrichten an Benutzer des UserInterfaces
:SEND_SNMP_ TRAP	Sendet einen SNMP-Trap
:SET	Versorgt Script-Variablen mit einem Wert
:SET_CALE	Ordnet ein Datum dem Kalenderbegriff eines Kalenders zu
:SET_CONDITION	Setzt die früheste Startzeit in Workflows
:SET_LAST_ERR	Setzt Fehlernummer und -text
:SET_SCRIPT_VAR	Setzt die Werte von Script-Variablen per indirektem Zugriff
:SET_UC_SETTING	Ändert Systemeinstellungen im laufenden Betrieb
:SHUTDOWN	Beendet ein AE-System
:STOP	Bricht die Bearbeitung eines Scripts ab
:SWITCH ... :CASE ... :ENDSWITCH	Überprüft, ob der Wert einer Variable bestimmten Werten entspricht und führt abhängig davon unterschiedliche Anweisungen durch.
:TERMINATE	Beendet einen Agenten, einen Arbeits- oder Kommunikationsprozess
:WAIT	Unterbricht die Scriptbearbeitung für einen festgelegten Zeitraum
:WHILE... :ENDWHILE	Schleife zur mehrfachen Ausführung von Scriptteilen
:XML_CLOSE	Schließt ein XML-Dokument

Script-Funktionen

[Script-Anweisungen] [Script-Funktionen]

Script-Funktion	Beschreibung	Vordefinierte Variable
ACTIVATE_UC_OBJECT	Aktiviert ein Objekt	
ADD	Führt eine Addition aus	
ADD_DAYS	Addiert Tage zu einem angegebenen Datum	
ADD_PERIOD	Addiert eine Periode zu einem angegebenen Datum	
ADD_TIME	Addiert zwei Zeiten	
ADD_TIMESTAMP	Addiert Zeit zu einem Zeitstempel	
ALPHA2RUNNR	Konvertiert den Namen einer Job bzw. Reportdatei in eine RunID	
ARRAY_2_STRING	Wandelt ein Script-Array in eine Zeichenkette um.	
AUTOFORECAST	Berechnet Prognosedaten für zukünftige Aktivitäten.	
CALE_LOOK_AHEAD	Ermittelt das nächste Datum aufgrund von Kalenderbedingungen.	
CANCEL_UC_OBJECT	Bricht ein aktiviertes Objekt ab	
CHANGE_LOGGING	Veranlasst einen Wechsel der Log-Datei.	
CONV_DATE	Konvertiert ein Datum von einem Datumsformat in ein anderes	
CONV_LC	Setzt alle Großbuchstaben einer Zeichenkette auf Kleinbuchstaben um	
CONV_TIMESTAMP	Rechnet Datum und Uhrzeit für eine andere Zeitzone um	
CONV_UC	Setzt alle Kleinbuchstaben einer Zeichenkette auf Großbuchstaben um	
CONVERT	Wandelt den Datentyp eines Wertes um.	
CREATE_OBJECT	Erstellt ein Objekt (nur Kalender, Login und Variable)	
CREATE_PROCESS	Erstellt ein neue Datensequenz.	
DAY_OF_YEAR	Liefert den laufenden Tag im Jahr	
DEACTIVATE_UC_OBJECT	Deaktiviert eine abgeschlossene Aufgabe.	
DIFF_DATE	Ermittelt die Differenz zwischen zwei Datumsangaben in Tagen	

DIFF_TIME	Ermittelt die Differenz zwischen zwei Zeitangaben	
DIV	Führt eine Division aus	
EXPORT	Exportiert Objekte in eine XML-Datei	
FIND	Durchsucht ein Script-Array und liefert den entsprechenden Index zurück.	
FIRST_OF_PERIOD	Ermittelt den ersten Tag der Periode für ein angegebenes Datum	
FORECAST_OBJECT	Erstellt Prognose für ein angegebenes Objekt	
FORECAST_TASK	Erstellt Prognose für eine laufende Aufgabe	
FORMAT	Änderung der Formatierung einer Zahl	
GET_ATT	Liefert den Wert von Attributen eines Objektes während der Generierung	
GET_ATT_PLAIN	Liefert den Wert von Attributen einer Aufgabe während der Generierung, wobei Variablen nicht aufgelöst werden	
GET_ATT_SUBSTR	Liefert einen Teil des Meldungstextes einer Benachrichtigung	
GET_BIT	Prüft, ob in einem Bit-Feld ein bestimmtes Bit gesetzt ist	
GET_CONDITION	Ermittelt die früheste Startzeit in Workflows	
GET_CONNECTION	Liest Informationen aus einem DB Verbindungs-Objekt.	
GET_CONSOLE	Liest Meldungsdaten des eingetretenen Konsolen-Ereignisses aus	
GET_EVENT_INFO	Liest Daten von eingetretenen Konsolen-, Dateisystem- und Datenbank-Ereignissen aus	
GET_FILESYSTEM	Ermittelt von einem Rechner ab einem definierten Pfad verschiedene Dateisystemwerte	
GET_LOGIN	Liest Informationen aus Login-Objekten.	
GET_MSG_TXT	Ermittelt den Meldungstext des zuletzt aufgetretenen Fehlers	
GET_MSG_TYPE	Ermittelt den Typ einer Meldungsnummer.	
GET_OBJECT_TYPE	Liefert den Objekttyp der Aufgabe	
GET_OH_IDNR	Liefert die interne Nummer eines Objektes	
GET_PARENT_NAME	Gibt den Namen der übergeordneten Aufgabe (Parent) zurück	
GET_PARENT_NR	Gibt die laufende Nummer (RunID) der übergeordneten Aufgabe (Parent) zurück	
GET_PARENT_TYPE	Gibt den Objekttyp der übergeordneten Aufgabe (Parent) zurück	
GET_PROCESS_INFO	Informationen einer Datensequenz ermitteln.	

GET_PROCESS_LINE	Ermittelt den aktuellen Zeileninhalt einer Datensequenz	
GET_PUBLISHED_VALUE	Wert einer Objekt- bzw. PromptSet-Variablen einer bestimmten Aufgabe ermitteln.	
GET_SCRIPT_VAR	Liefert die Werte von Script-Variablen per indirektem Zugriff	
GET_STATISTIC_DETAIL	Ermittelt Details aus einem Statistiksatz eines aktivierbaren Objektes	
GET_SYNC	Fragt den aktuellen Zustand oder Wert eines Sync-Objektes ab	
GET_UC_OBJECT_NR	Liefert die RunID eines aktivierten Objektes	
GET_UC_OBJECT_STATUS	Liefert den Status eines aktivierten Objektes	
GET_UC_SERVER_NAME	Ermittelt den Namen des Arbeitsprozesses, in dem das Script läuft	
GET_UC_SETTING	Liest aktuelle Systemeinstellungen	
GET_UC_SYSTEM_NAME	Ermittelt den Namen des AE-Systems	&$SYSTEM#
GET_VAR	Liefert den Wert eines Variablen-Objektes.	
GET_WIN_EVENT	Ermittelt Einträge in das System-, Sicherheits- und Anwendungsprotokoll von Windows NT, 2000 oder XP beim Eintreten eines Ereignisses	
HEX	Konvertiert eine Zeichenkette in hexadezimale Schreibweise	
ILM	Steuert die ILM-Funktionalität	
IMPORT	Importiert Objekte aus einer XML-Datei	
IS_GROUP_MEMBER	Prüft, ob ein Benutzer Mitglied in einer BenutzerGruppe ist	
ISNUMERIC	Prüft, ob eine Zeichenkette numerisch ist	
LAST_OF_PERIOD	Ermittelt den letzten Tag der Periode für ein angegebenes Datum	
LENGTH	Ermittelt die Größe eines Script-Arrays.	
LOAD_PROCESS	Lädt eine gespeicherte Datensequenz.	
MID	Kopiert Zeichen aus einer Zeichenkette	
MOD	Liefert den Restwert einer Division	
MODIFY_OBJECT	Ändert ein vorhandenes Objekt (nur Kalender, Login und Variable)	
MODIFY_SYSTEM	Führt ServiceManager-Aktionen oder Queue-Modifikationen durch.	
MODIFY_TASK	Ändert laufende Workflows.	
MODIFY_UC_OBJECT	Ändert das Attribut eines aktivierten Objektes	
MOVE_OBJECT	Verschiebt ein Objekt in einen Ordner	

MULT	Führt eine Multiplikation aus	
PREP_PROCESS	Führt mit Hilfe von speziellen Job-Objekten (Ereignis-Jobs), Kommandos auf einem Rechner aus, und liefert die Konsolen-Ausgabe als interne Auflistung (Datensequenz), die zur Weiterverarbeitung genutzt werden kann, zurück.	
PREP_PROCESS_AGENTGROUP	Ermittelt mit Auswahlkriterien die Agenten eines AgentenGruppen-Objektes und stellt das Ergebnis als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.	
PREP_PROCESS_COMMENTS	Ermittelt mit Filtereinstellungen den Zeitstempel, Benutzer und Text von Aufgaben-Kommentaren und stellt das Ergebnis als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.	
PREP_PROCESS_DOCU	Stellt den Inhalt einer Dokumentations-Registerkarte als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.	
PREP_PROCESS_FILE	Ermittelt mit Filterkriterien zeilenweise den Inhalt einer Textdatei, die sich auf einem angegebenen Rechner befindet, und stellt das Ergebnis als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.	
PREP_PROCESS_FILENAME	Ermittelt eine Liste mit Namen von Dateien, die sich in dem angegebenen Verzeichnis eines Rechners befinden, und stellt diese als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.	
PREP_PROCESS_PROMPTSET	Liest die Definition von PromptSet-Objekten aus und stellt diese in Form einer internen Auflistung (Datensequenz) für die weitere Verarbeitung bereit.	
PREP_PROCESS_REPORT	Ermittelt mit Filterkriterien Reportzeilen von aktivierbaren Objekten und stellt das Ergebnis als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.	
PREP_PROCESS_REPORTLIST	Ermittelt die Liste des registrierten Outputs von bereits gelaufenen Jobs und stellt das Ergebnis als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.	
PREP_PROCESS_VAR	Ermittelt mit Auswahlkriterien eine Liste von Werten eines Variablenobjektes und stellt diese als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.	
PUT_PROCESS_LINE	Fügt einer bestimmten Datensequenz eine Zeile hinzu.	

RANDOM	Generiert Zufallszahlen	
REMOVE_OBJECT	Löscht ein vorhandenes Objekt	
RERUN_UC_OBJECT	Fortsetzen eines bestimmten Workflows.	
RESTART_UC_ OBJECT	Wiederholt die Durchführung einer Aufgabe	
ROLLBACK_UC_ OBJECT	Rollback einer bestimmten Aufgabe durchführen.	
RUNNR2ALPHA	Konvertiert RunID in den zugehörigen Dateinamen	
SAVE_PROCESS	Speichert eine bestimmte Datensequenz.	
SEND_MAIL	Sendet E-Mail an einen Benutzer	
SET_SYNC	Führt die definierte Aktion eines Sync-Objektes aus	
STR_CAT	Verbindet zwei Zeichenketten zu einer neuen Zeichenkette	
STR_CUT	Kopiert Zeichen aus einer Zeichenkette	
STR_ENDS_WITH	Prüft ob eine Zeichenkette mit einer bestimmten anderen Zeichenkette endet.	
STR_FIND	Sucht ein Zeichen oder eine Zeichenkette innerhalb einer Zeichenkette	
STR_FIND_REVERSE	Sucht ein Zeichen oder eine Zeichenkette innerhalb einer Zeichenkette. Die Suche beginnt am Ende der zu durchsuchenden Zeichenkette.	
STR_ISLOWER	Prüft die Buchstaben einer Zeichenkette auf Kleinschreibung.	
STR_ISUPPER	Prüft die Buchstaben einer bestimmten Zeichenkette auf Großschreibung.	
STR_LC	Setzt alle Großbuchstaben einer Zeichenkette auf Kleinbuchstaben um	
STR_LENGTH	Ermittelt die Länge einer Zeichenkette	
STR_LNG	Ermittelt die Länge einer Zeichenkette	
STR_LTRIM	Entfernt Leerzeichen am Anfang einer Zeichenkette	
STR_MATCH	Vergleicht zwei Zeichenketten	
STR_PAD	Zeichenkette auf eine bestimmte Länge erweitern.	
STR_REVERSE	Kehrt die Reihenfolge der Zeichen einer Zeichenkette um	
STR_RTRIM	Entfernt Leerzeichen am Ende einer Zeichenkette	
STR_SPLIT	Zeichenkette mit Hilfe eines Trennzeichens in mehrere Teile aufspalten.	

STR_STARTS_WITH	Prüft ob eine Zeichenkette mit einer bestimmten anderen Zeichenkette beginnt.	
STR_SUBSTITUTE	Ersetzt Zeichen oder Zeichenkette innerhalb einer Zeichenkette	
STR_SUBSTITUTE_VAR	Ersetzt den Namen von Script-Variablen durch deren Wert.	
STR_TRIM	Entfernt Leerzeichen am Anfang und Ende einer Zeichenkette	
STR_UC	Setzt alle Kleinbuchstaben einer Zeichenkette auf Großbuchstaben um	
SUB	Führt eine Subtraktion aus	
SUB_DAYS	Subtrahiert Tage von einem angegebenen Datum	
SUB_PERIOD	Subtrahiert eine Periode von einem angegebenen Datum	
SUB_TIME	Subtrahiert zwei Zeiten	
SUB_TIMESTAMP	Subtrahiert Zeit von einem Zeitstempel	
SUBSTR	Kopiert Zeichen aus einer Zeichenkette	
SYS_ACT_CLIENT	Liefert die Nummer des aktuellen Mandanten	&$CLIENT#
SYS_ACT_CLIENT_TEXT	Liefert den Text des aktuellen Mandanten	&$CLIENT_DESC#
SYS_ACT_HOST	Ermittelt den Namen des Hosts	
SYS_ACT_JP	Ermittelt, ob die Aufgabe aus einem Workflow heraus aktiviert wurde	&$IN_PROCESSFLOW#
SYS_ACT_ME_NAME	Liefert den Namen des eigenen Objektes	&$NAME#
SYS_ACT_ME_NR	Liefert die laufende Nummer (RunID) des eigenen Objektes	&$RUNID#
SYS_ACT_ME_TYPE	Liefert den Objekttyp des eigenen Objektes	&$OBJECT_TYPE#
SYS_ACT_PARENT_NAME	Liefert den Namen der übergeordneten Aufgabe	&$ACTIVATOR# &$PROCESSOR#
SYS_ACT_PARENT_NR	Liefert die laufende Nummer (RunID) der übergeordneten Aufgabe	&$ACTIVATOR_RUNID# &$PROCESSOR_RUNID#
SYS_ACT_PARENT_TYPE	Liefert den Objekttyp der übergeordneten Aufgabe	&$ACTIVATOR_TYPE# &$PROCESSOR_TYPE#
SYS_ACT_PREV_NAME	Liefert den Namen der vorhergehenden Aufgabe in einem Workflow	
SYS_ACT_PREV_NR	Liefert die laufende Nummer (RunID) der vorhergehenden Aufgabe in einem Workflow	

SYS_ACT_PTTYP	Liefert Partnertyp des Benutzers	&$PARTNER_TYPE#
SYS_ACT_RESTART	Ermittelt, ob Objekt im Wiederanlauf-Modus aktiviert wurde	&$RESTARTED#
SYS_ACT_RESTART_COUNT	Liefert die Anzahl der Wiederanläufe von Workflow-Aufgaben, die über die Anweisung RESTART TASK durchgeführt wurden (Post-Conditions)	&$RESTART_COUNT#
SYS_ACT_RESTART_ME_NR	Liefert die laufende Nummer (RunID) des im Wiederanlauf-Modus aktivierten Objektes	&$RESTART_RUNID#
SYS_ACT_TOP_NAME	Liefert den Namen des obersten Workflows	&$TOP_PROCESSFLOW_NAME#
SYS_ACT_TOP_NR	Ermittelt die laufende Nummer (RunID) des obersten Workflows	&$TOP_PROCESSFLOW_RUNID#
SYS_ACT_USERID	Liefert die Benutzerkennung, unter welcher der Job laufen wird	
SYS_ACTIVE_COUNT	Ermittelt Anzahl der in den Aktivitäten stehenden Objekte	
SYS_BUSY_01	Liefert die prozentuale Auslastung der Automation Engine während der letzten Minute	
SYS_BUSY_10	Liefert die prozentuale Auslastung der Automation Engine während der letzten 10 Minuten	
SYS_BUSY_60	Liefert die prozentuale Auslastung der Automation Engine während der letzten Stunde	
SYS_DATE	Ermittelt das aktuelle Tagesdatum zu Beginn der Script-Prozessierung	&$DATE_*format*#
SYS_DATE_PHYSICAL	Ermittelt das jeweils aktuelle Tagesdatum	&$PHYS_DATE_*format*#
SYS_HOST_ALIVE	Überprüft, ob ein bestimmter Host aktiv ist	
SYS_INFO	Liest Informationen über das AE-System aus	
SYS_LAST_ERR_INS	Liefert den variablen Teil der Fehlermeldung bei einem zuletzt aufgetretenen Fehler	
SYS_LAST_ERR_NR	Gibt die Fehlernummer eines zuletzt aufgetretenen Fehlers zurück	
SYS_LAST_ERR_SYSTXT	Ermittelt den Fehlertext des Betriebssystems eines zuletzt aufgetretenen Fehlers	
SYS_LAST_RESTART_POINT	Liefert die Bezeichnung des vorangegangenen Wiederanlauf-Punktes im Script	
SYS_LAST_RESTART_TEXT	Liefert den Text des vorangegangenen Wiederanlauf-Punktes im Script	
SYS_LDATE	Ermittelt das logische Datum	&$LDATE_*format*#

SYS_RESTART_ POINT	Liefert den Wiederanlauf-Punkt, mit dem das Objekt ausgeführt wird	&$RESTART_ POINT#
SYS_SERVER_ALIVE	Überprüft, ob ein bestimmter Serverprozess aktiv ist	
SYS_SNMP_ACTIVE	Prüft, ob die SNMP-Kopplung (Simple Network Management Protocol) der AE aktiv ist	&$SNMP_ ACTIVE#
SYS_STATE_ACTIVE	Prüft, ob ein Objekt bereits aktiv ist	
SYS_STATE_JOB_ ACTIVE	Prüft, ob ein Job bereits aktiv ist	
SYS_STATE_JOBS_ IN_GROUP	Ermittelt die Anzahl von Jobs, die in Gruppen vorgemerkt sind	
SYS_STATE_JP_ ACTIVE	Prüft, ob ein Workflow bereits aktiv ist	
SYS_TIME	Stellt die aktuelle Tageszeit zu Beginn der Script-Prozessierung fest	&$TIME_*format*#
SYS_TIME_PHYSICAL	Stellt die jeweils aktuelle Tageszeit fest	&$PHYS_TIME_ *format*#
SYS_TIMESTAMP_ PHYSICAL	Liefert aktuelles Datum und Uhrzeit	
SYS_USER_ALIVE	Prüft, ob der Benutzer mit einem UserInterface an der Automation Engine angemeldet ist	
SYS_USER_DEP	Liefert die Abteilung des Benutzers, der die Aufgabe startete	&$DEPARTMENT#
SYS_USER_ LANGUAGE	Liefert die Sprache in der der Server die Logdateien erzeugt	&$SYS_ LANGUAGE#
SYS_USER_LNAME	Liefert den Vor- und Nachnamen des Benutzers, der die Aufgabe startete	&$USER_FL#
SYS_USER_NAME	Liefert den Namen des Benutzers, der die Aufgabe startete	&$USER#
TOGGLE_OBJECT_ STATUS	Stoppt oder startet die automatische Verarbeitung von manchen Objekttypen	
TOGGLE_SYSTEM_ STATUS	Stoppt oder startet die automatische Verarbeitung eines gesamten Mandanten	
UC_CRLF	Liefert einen Zeilenumbruch	
VALID_CALE	Prüft, ob ein Datum in einem Kalenderbegriff enthalten ist	
VALID_DATE	Prüft, ob ein Datum gültig ist	
VALID_TIME	Prüft, ob eine Zeit gültig ist	
WEEK_NR	Liefert die Kalenderwoche, in der sich ein Datum befindet	
WEEKDAY_NR	Liefern den Wochentag eines Datums als Ziffer	
WEEKDAY_XX	Gibt den Wochentag als Kürzel zurück, der für ein Datum gilt	

WRITE_PROCESS	Schreibt den Inhalt einer Datensequenz in eine Datei.	
XML_BEAUTIFY	Bereitet die Darstellung einer Elementstruktur auf	
XML_GET_ATTRIBUTE	Liefert den Wert eines Attributes	
XML_GET_CHILD_COUNT	Zählt Subelemente eines Elementes	
XML_GET_FIRST_CHILD	Ermittelt das erste Subelement eines Elementes	
XML_GET_LAST_CHILD	Ermittelt das letzte Subelement eines Elementes	
XML_GET_NEXTSIBLING	Ermittelt das nachfolgende Element	
XML_GET_NODE_NAME	Liefert den Namen eines Elementes	
XML_GET_NODE_TEXT	Liefert den Text eines Elementes	
XML_OPEN	Öffnet ein XML-Dokument für die Bearbeitung	
XML_PRINTINTOFILE	Gibt das XML-Dokument in eine Datei aus	
XML_SELECT_NODE	Ermittelt beliebiges Element	
YEAR_9999	Extrahiert das Jahr aus einem angegebenen Datum	

3 Funktionelle Gliederung

3.1 Script-Sprachmittel - funktionelle Gliederung

Folgende Übersicht stellt die Scriptprachmittel geordnet nach ihren Funktionsbereichen dar.

Objekte bearbeiten

[Objekte bearbeiten] [Objekte aktivieren] [Objekte lesen oder verändern] [Script-Aufbau und -Prozessierung] [Fehlerbehandlung und Meldungen] [Aktivierungsdaten] [Benutzerdaten] [Datensequenzen] [Ereignisbehandlung] [Systemzustände und -bedienung] [Datum und Zeit] [Rechnen] [Strings]

Script-Sprachmittel	Beschreibung
:REGISTER_ OUTPUTFILE	Registriert eine Datei als externen Job-Output
CREATE_OBJECT	Erstellt ein Objekt (nur Kalender, Login und Variable)
EXPORT	Exportiert Objekte in eine XML-Datei
IMPORT	Importiert Objekte aus einer XML-Datei
MODIFY_OBJECT	Ändert ein vorhandenes Objekt (nur Kalender, Login und Variable)
MOVE_OBJECT	Verschiebt ein Objekt in einen Ordner
REMOVE_OBJECT	Löscht ein vorhandenes Objekt

Objekte aktivieren

[Objekte bearbeiten] [Objekte aktivieren] [Objekte lesen oder verändern] [Script-Aufbau und -Prozessierung] [Fehlerbehandlung und Meldungen] [Aktivierungsdaten] [Benutzerdaten] [Datensequenzen] [Ereignisbehandlung] [Systemzustände und -bedienung] [Datum und Zeit] [Rechnen] [Strings]

Script-Sprachmittel	Beschreibung
:BEGINREAD... :ENDREAD	Beginn und Ende eines Dialogfeldes für Benutzerabfragen
:PRINT	Gibt Text in einem Dialogfeld für Benutzerabfragen oder im Aktivierungsprotokoll eines Objektes aus
:PUT_READ_ BUFFER	Stellt Namen und Inhalt einer Script-Variablen in einen Zwischenspeicher
:PUT_PROMPT_ BUFFER	Stellt Namen und Inhalt einer Script-Variablen in einen Zwischenspeicher
:READ	Fragt Angaben vom Benutzer im Dialog ab

ACTIVATE_UC_OBJECT	Aktiviert ein Objekt
AUTOFORECAST	Berechnet Prognosedaten für zukünftige Aktivitäten.
CANCEL_UC_OBJECT	Bricht ein aktiviertes Objekt ab
DEACTIVATE_UC_OBJECT	Deaktiviert eine abgeschlossene Aufgabe.
FORECAST_OBJECT	Erstellt Prognose für ein angegebenes Objekt
FORECAST_TASK	Erstellt Prognose für eine laufende Aufgabe
RERUN_UC_OBJECT	Fortsetzen eines bestimmten Workflows.
RESTART_UC_OBJECT	Wiederholt die Durchführung einer Aufgabe
ROLLBACK_UC_OBJECT	Rollback einer bestimmten Aufgabe durchführen.
SYS_ACTIVE_COUNT	Ermittelt Anzahl der in den Aktivitäten stehenden Objekte
SYS_STATE_ACTIVE	Prüft, ob ein Objekt bereits aktiv ist
SYS_STATE_JOB_ACTIVE	Prüft, ob ein Job bereits aktiv ist
SYS_STATE_JOBS_IN_GROUP	Ermittelt die Anzahl von Jobs, die in Gruppen vorgemerkt sind
SYS_STATE_JP_ACTIVE	Prüft, ob ein Workflow bereits aktiv ist
TOGGLE_OBJECT_STATUS	Stoppt oder startet die automatische Verarbeitung von manchen Objekttypen

Objekte lesen oder verändern

[Objekte bearbeiten] [Objekte aktivieren] [Objekte lesen oder verändern] [Script-Aufbau und -Prozessierung] [Fehlerbehandlung und Meldungen] [Aktivierungsdaten] [Benutzerdaten] [Datensequenzen] [Ereignisbehandlung] [Systemzustände und -bedienung] [Datum und Zeit] [Rechnen] [Strings]

Script-Sprachmittel	Beschreibung
:ADD_ATT	Fügt einem Benachrichtigungs-Objekt Empfänger zur Laufzeit hinzu
:ADD_COMMENT	Fügt einer Aufgabe einen Kommentar hinzu
:ATTACH_SYNC	Weist einer Aufgabe ein Sync-Objekt zu
:DELETE_VAR	Löscht einen oder alle Werte aus einem statichen Variablen-Objekt
:MODIFY_STATE	Ändert Rückgabewert oder Statustext eines Jobs nach dessen Ende
:PUT_ATT	Setzt oder ändert Attribute von Objekten

:PUT_ATT_APPEND	Erweitert den Meldungstext einer Benachrichtigung zur Laufzeit
:PUT_VAR	Speichert Werte in ein statisches Variablen-Objekt
:PUT_VAR_COL	Speichert einen Wert in eine bestimmte Spalte eines statischen Variablen-Objektes
:REMOVE_ATT	Entfernt in einem Benachrichtigungs-Objekt Empfänger zur Laufzeit
:REPLACE_ STRUCTURE	Ersetzt die Struktur eines Workflows bei dessen Aktivierung durch die Struktur eines anderen Workflows.
:SET_CALE	Ordnet ein Datum dem Kalenderbegriff eines Kalenders zu
:SET_CONDITION	Setzt die früheste Startzeit in Workflows
:XML_CLOSE	Schließt ein XML-Dokument
GET_ATT	Liefert den Wert von Attributen eines Objektes während der Generierung
GET_ATT_PLAIN	Liefert den Wert von Attributen einer Aufgabe während der Generierung, wobei Variablen nicht aufgelöst werden.
GET_ATT_SUBSTR	Liefert einen Teil des Meldungstextes einer Benachrichtigung
GET_CONDITION	Ermittelt die früheste Startzeit in Workflows
GET_CONNECTION	Liest Informationen aus einem DB Verbindungs-Objekt.
GET_LOGIN	Liest Informationen aus Login-Objekten.
GET_OBJECT_ TYPE	Liefert den Objekttyp der Aufgabe
GET_OH_IDNR	Liefert die interne Nummer eines Objektes
GET_PUBLISHED_ VALUE	Wert einer Objekt- bzw. PromptSet-Variablen einer bestimmten Aufgabe ermitteln.
GET_STATISTIC_ DETAIL	Ermittelt Details aus einem Statistiksatz eines aktivierbaren Objektes
GET_SYNC	Fragt den aktuellen Zustand oder Wert eines Sync-Objektes ab
GET_VAR	Liefert den Wert eines Variablen-Objektes.
MODIFY_TASK	Ändert laufende Workflows
MODIFY_UC_ OBJECT	Ändert das Attribut eines aktivierten Objektes
SET_SYNC	Führt die definierte Aktion eines Sync-Objektes aus
XML_BEAUTIFY	Bereitet die Darstellung einer Elementstruktur auf
XML_GET_ ATTRIBUTE	Liefert den Wert eines Attributes
XML_GET_CHILD_ COUNT	Zählt Subelemente eines Elementes
XML_GET_FIRST_ CHILD	Ermittelt das erste Subelement eines Elementes
XML_GET_LAST_ CHILD	Ermittelt das letzte Subelement eines Elementes
XML_GET_ NEXTSIBLING	Ermittelt das nachfolgende Element

XML_GET_NODE_NAME	Liefert den Namen eines Elementes
XML_GET_NODE_TEXT	Liefert den Text eines Elementes
XML_OPEN	Öffnet ein XML-Dokument für die Bearbeitung
XML_PRINTINTOFILE	Gibt das XML-Dokument in eine Datei aus
XML_SELECT_NODE	Ermittelt beliebiges Element

Script-Aufbau und -Prozessierung

[Objekte bearbeiten] [Objekte aktivieren] [Objekte lesen oder verändern] [Script-Aufbau und -Prozessierung] [Fehlerbehandlung und Meldungen] [Aktivierungsdaten] [Benutzerdaten] [Datensequenzen] [Ereignisbehandlung] [Systemzustände und -bedienung] [Datum und Zeit] [Rechnen] [Strings]

Script-Sprachmittel	Beschreibung
:CLEAR	Setzt ein Script-Array wieder auf seine Initialwerte zurück.
:CONST	Erstellt eine Script-Variable als Konstante mit einem bestimmten Wert.
:DATA	Ausdrückliche Deklaration einer DATA-Zeile im Script
:DEFINE	Erstellung einer Script-Variable mit einem bestimmten Datentyp.
:FILL	Speichert mehrere Werte in ein Script-Array.
:EXT_REPORT_OFF	Schaltet das Protokollieren des Scripts einer Aufgabe aus
:EXT_REPORT_ON	Schaltet das Protokollieren des Scripts einer Aufgabe ein
:GENERATE	Steuert die Behandlung von Script-Zeilen bei der Prozessierung eines Scripts
:IF... :ELSE... :ENDIF	Verzweigung in Abhängigkeit von Bedingungen
:INCLUDE	Bindet ein Include-Objekt in das aktuelle Script ein
:INC_SCRIPT	Bindet ein Script in ein anderes Script des gleichen Objektes ein
:JCL_CONCAT_CHAR	Bildet JCL-Zeilen bis zu einer maximalen Länge von 2 KB
:JCL_SUBSTITUTE	Ersetzt eine Zeichenkette in der JCL durch eine andere Zeichenkette
:PSET	Weist einer Objektvariablen einen Wert zu.
:PUBLISH	Script-Variablen und Arrays als Objektvariablen definieren.
:RESTART	Setzt Restartpunkte in einem Objekt
:RSET	Weist einer Script-Variablen einen Wert zu und speichert diesen im Aktivierungsreport
:SET	Versorgt Script-Variablen mit einem Wert
:SET_SCRIPT_VAR	Setzt die Werte von Script-Variablen per indirektem Zugriff

:SWITCH ... :CASE ... :ENDSWITCH	Überprüft, ob der Wert einer Variable bestimmten Werten entspricht und führt abhängig davon unterschiedliche Anweisungen durch.
:WAIT	Unterbricht die Scriptbearbeitung für einen festgelegten Zeitraum
:WHILE... :ENDWHILE	Schleife zur mehrfachen Ausführung von Scriptteilen
FIND	Durchsucht ein Script-Array und liefert den entsprechenden Index zurück.
GET_SCRIPT_VAR	Liefert die Werte von Script-Variablen per indirektem Zugriff
LENGTH	Ermittelt die Größe eines Script-Arrays.

Fehlerbehandlung und Meldungen

[Objekte bearbeiten] [Objekte aktivieren] [Objekte lesen oder verändern] [Script-Aufbau und -Prozessierung] [Fehlerbehandlung und Meldungen] [Aktivierungsdaten] [Benutzerdaten] [Datensequenzen] [Ereignisbehandlung] [Systemzustände und -bedienung] [Datum und Zeit] [Rechnen] [Strings]

Script-Sprachmittel	Beschreibung
:EXIT	Beendet die Prozessierung des Scripts mit Rückgabewert
:ON_ERROR	Legt die Reaktion auf bestimmte Fehler bzw. Meldungen von Script-Sprachmitteln fest
:SEND_MSG	Schickt Nachrichten an Benutzer des UserInterfaces
:SEND_SNMP_TRAP	Sendet einen SNMP-Trap
:SET_LAST_ERR	Setzt Fehlernummer und -text
:STOP	Beendet die Aktivierung eines Objektes und kann Fehlermeldung ausgeben
GET_MSG_TXT	Ermittelt den Meldungstext des zuletzt aufgetretenen Fehlers
GET_MSG_TYPE	Ermittelt den Typ einer Meldungsnummer.
SEND_MAIL	Sendet E-Mails an einen Benutzer
SYS_LAST_ERR_INS	Liefert den variablen Teil der Fehlermeldung bei einem zuletzt aufgetretenen Fehler
SYS_LAST_ERR_NR	Gibt die Fehlernummer eines zuletzt aufgetretenen Fehlers zurück
SYS_LAST_ERR_SYSTXT	Ermittelt den Fehlertext des Betriebssystems eines zuletzt aufgetretenen Fehlers

Aktivierungsdaten

[Objekte bearbeiten] [Objekte aktivieren] [Objekte lesen oder verändern] [Script-Aufbau und -Prozessierung] [Fehlerbehandlung und Meldungen] [Aktivierungsdaten] [Benutzerdaten]

[Datensequenzen] [Ereignisbehandlung] [Systemzustände und -bedienung] [Datum und Zeit] [Rechnen] [Strings]

Script-Sprachmittel	Beschreibung	Vordefinierte Variable
GET_PARENT_NAME	Gibt den Namen der übergeordneten Aufgabe (Parent) zurück	
GET_PARENT_NR	Gibt die laufende Nummer (RunID) der übergeordneten Aufgabe (Parent) zurück	
GET_PARENT_TYPE	Gibt den Objekttyp der übergeordneten Aufgabe (Parent) zurück	
GET_UC_OBJECT_NR	Liefert die RunID eines aktivierten Objektes	
GET_UC_OBJECT_STATUS	Liefert den Status eines aktivierten Objektes	
SYS_ACT_HOST	Ermittelt den Namen des Hosts	
SYS_ACT_JP	Ermittelt, ob die Aufgabe aus einem Workflow heraus aktiviert wurde	&$IN_ PROCESSFLOW#
SYS_ACT_ME_NAME	Liefert den Namen des eigenen Objektes	&$NAME#
SYS_ACT_ME_NR	Liefert die laufende Nummer (RunID) des eigenen Objektes	&$RUNID#
SYS_ACT_ME_TYPE	Liefert den Objekttyp des eigenen Objektes	&$OBJECT_ TYPE#
SYS_ACT_PARENT_NAME	Liefert den Namen der übergeordneten Aufgabe	&$ACTIVATOR# &$PROCESSOR#
SYS_ACT_PARENT_NR	Liefert die laufende Nummer (RunID) der übergeordneten Aufgabe	&$ACTIVATOR_ RUNID# &$PROCESSOR_ RUNID#
SYS_ACT_PARENT_TYPE	Liefert den Objekttyp der übergeordneten Aufgabe	&$ACTIVATOR_ TYPE# &$PROCESSOR_ TYPE#
SYS_ACT_PREV_NAME	Liefert den Namen der vorhergehenden Aufgabe in einem Workflow	
SYS_ACT_PREV_NR	Liefert die laufende Nummer (RunID) der vorhergehenden Aufgabe in einem Workflow	
SYS_ACT_PTTYP	Liefert Partnertyp des Benutzers	&$PARTNER_ TYPE#
SYS_ACT_RESTART	Ermittelt, ob Objekt im Wiederanlauf-Modus aktiviert wurde	&$RESTARTED#

SYS_ACT_RESTART_COUNT	Liefert die Anzahl der Wiederanläufe von Workflow-Aufgaben, die über die Anweisung RESTART TASK durchgeführt wurden (Post-Conditions)	&$RESTART_COUNT#
SYS_ACT_RESTART_ME_NR	Liefert die laufende Nummer (RunID) des im Wiederanlauf-Modus aktivierten Objektes	&$RESTART_RUNID#
SYS_ACT_TOP_NAME	Liefert den Namen des obersten Workflows	&$TOP_PROCESSFLOW_NAME#
SYS_ACT_TOP_NR	Ermittelt die laufende Nummer (RunID) des obersten Workflows	&$TOP_PROCESSFLOW_RUNID#
SYS_ACT_USERID	Liefert die Benutzerkennung, unter welcher der Job laufen wird	
SYS_LAST_RESTART_POINT	Liefert die Bezeichnung des vorangegangenen Wiederanlauf-Punktes im Script	
SYS_LAST_RESTART_TEXT	Liefert den Text des vorangegangenen Wiederanlauf-Punktes im Script	
SYS_RESTART_POINT	Liefert den Wiederanlauf-Punkt, mit dem das Objekt ausgeführt wird	&$RESTART_POINT#

Benutzerdaten

[Objekte bearbeiten] [Objekte aktivieren] [Objekte lesen oder verändern] [Script-Aufbau und -Prozessierung] [Fehlerbehandlung und Meldungen] [Aktivierungsdaten] [Benutzerdaten] [Datensequenzen] [Ereignisbehandlung] [Systemzustände und -bedienung] [Datum und Zeit] [Rechnen] [Strings]

Script-Sprachmittel	Beschreibung	Vordefinierte Variable
IS_GROUP_MEMBER	Prüft, ob ein Benutzer Mitglied in einer BenutzerGruppe ist	
SYS_ACT_CLIENT	Liefert die Nummer des aktuellen Mandanten	&$CLIENT#
SYS_ACT_CLIENT_TEXT	Liefert den Text des aktuellen Mandanten	&$CLIENT_DESC#
SYS_USER_ALIVE	Prüft, ob der Benutzer mit einem UserInterface an die Automation Engine angemeldet ist	
SYS_USER_DEP	Liefert die Abteilung des Benutzers, der die Aufgabe startete	&$DEPARTMENT#
SYS_USER_LNAME	Liefert den Vor- und Nachnamen des Benutzers, der die Aufgabe startete	&$USER_FL#
SYS_USER_NAME	Liefert den Namen des Benutzers, der die Aufgabe startete	&$USER#

Datensequenzen

[Objekte bearbeiten] [Objekte aktivieren] [Objekte lesen oder verändern] [Script-Aufbau und -Prozessierung] [Fehlerbehandlung und Meldungen] [Aktivierungsdaten] [Benutzerdaten] [Datensequenzen] [Ereignisbehandlung] [Systemzustände und -bedienung] [Datum und Zeit] [Rechnen] [Strings]

Script-Sprachmittel	Beschreibung
:CLOSE_PROCESS	Verwirft eine Datensequenz innerhalb eines Scripts
:PROCESS... :TERM_PROCESS... :ENDPROCESS	Schleife zur zeilenweisen Bearbeitung einer Datensequenz, beispielsweise sequentielle Datei oder Kommandoergebnis
CREATE_PROCESS	Erstellt ein neue Datensequenz.
GET_PROCESS_ INFO	Informationen einer Datensequenz ermitteln.
GET_PROCESS_ LINE	Ermittelt den aktuellen Zeileninhalt einer Datensequenz
LOAD_PROCESS	Lädt eine gespeicherte Datensequenz.
PREP_PROCESS	Führt mit Hilfe von speziellen Job-Objekten (Ereignis-Jobs), Kommandos auf einem Rechner aus, und liefert die Konsolen-Ausgabe als interne Auflistung (Datensequenz), die zur Weiterverarbeitung genutzt werden kann, zurück.
PREP_PROCESS_ AGENTGROUP	Ermittelt mit Auswahlkriterien die Agenten eines AgentenGruppen-Objektes und stellt das Ergebnis als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.
PREP_PROCESS_ COMMENTS	Ermittelt mit Filtereinstellungen den Zeitstempel, Benutzer und Text von Aufgaben-Kommentaren und stellt das Ergebnis als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.
PREP_PROCESS_ DOCU	Stellt den Inhalt einer Dokumentations-Registerkarte als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.
PREP_PROCESS_ FILE	Ermittelt mit Filterkriterien zeilenweise den Inhalt einer Textdatei, die sich auf einem angegebenen Rechner befindet, und stellt das Ergebnis als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.
PREP_PROCESS_ FILENAME	Ermittelt eine Liste mit Namen von Dateien, die sich in dem angegebenen Verzeichnis eines Rechners befinden, und stellt diese als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.
PREP_PROCESS_ PROMPTSET	Liest die Definition von PromptSet-Objekten aus und stellt diese in Form einer internen Auflistung (Datensequenz) für die weitere Verarbeitung bereit.
PREP_PROCESS_ REPORT	Ermittelt mit Filterkriterien Reportzeilen von aktivierbaren Objekten und stellt das Ergebnis als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.

PREP_PROCESS_REPORTLIST	Ermittelt die Liste des registrierten Outputs von bereits gelaufenen Jobs und stellt das Ergebnis als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.
PREP_PROCESS_VAR	Ermittelt mit Auswahlkriterien eine Liste von Werten eines Variablenobjektes und stellt diese als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.
PUT_PROCESS_LINE	Fügt einer bestimmten Datensequenz eine Zeile hinzu.
SAVE_PROCESS	Speichert eine bestimmte Datensequenz.
WRITE_PROCESS	Schreibt den Inhalt einer Datensequenz in eine Datei.

Ereignisbehandlung

[Objekte bearbeiten] [Objekte aktivieren] [Objekte lesen oder verändern] [Script-Aufbau und -Prozessierung] [Fehlerbehandlung und Meldungen] [Aktivierungsdaten] [Benutzerdaten] [Datensequenzen] [Ereignisbehandlung] [Systemzustände und -bedienung] [Datum und Zeit] [Rechnen] [Strings]

Script-Sprachmittel	Beschreibung
GET_CONSOLE	Liest Meldungsdaten des eingetretenen Konsolen-Ereignisses aus
GET_EVENT_INFO	Liest Daten von eingetretenen Konsolen-, Dateisystem- und Datenbank-Ereignissen aus
GET_FILESYSTEM	Ermittelt von einem Rechner ab einem definierten Pfad verschiedene Dateisystemwerte.
GET_WIN_EVENT	Ermittelt Einträge in das System-, Sicherheits- und Anwendungsprotokoll von Windows NT, 2000 oder XP beim Eintreten eines Ereignisses

3.1.1 Systemzustände und -bedienung

[Objekte bearbeiten] [Objekte aktivieren] [Objekte lesen oder verändern] [Script-Aufbau und -Prozessierung] [Fehlerbehandlung und Meldungen] [Aktivierungsdaten] [Benutzerdaten] [Datensequenzen] [Ereignisbehandlung] [Systemzustände und -bedienung] [Datum und Zeit] [Rechnen] [Strings]

Script-Sprachmittel	Beschreibung	Vordefinierte Variable
:DISCONNECT	Trennt eine Verbindung zum AE-System	
:SET_UC_SETTING	Ändert Systemeinstellungen im laufenden Betrieb	
:SHUTDOWN	Beendet ein AE-System	
:TERMINATE	Beendet einen Agenten, einen Arbeits- oder Kommunikationsprozess	
CHANGE_LOGGING	Veranlasst einen Wechsel der Log-Datei.	

GET_UC_SERVER_NAME	Ermittelt den Namen des Arbeitsprozesses, in dem das Script läuft	
GET_UC_SETTING	Liest aktuelle Systemeinstellungen	
GET_UC_SYSTEM_NAME	Ermittelt den Namen des AE-Systems	&$SYSTEM#
ILM	Steuert die ILM-Funktionalität	
MODIFY_SYSTEM	Führt ServiceManager-Aktionen oder Queue-Modifikationen durch.	
SYS_BUSY_01	Liefert die prozentuale Auslastung der Automation Engine während der letzten Minute	
SYS_BUSY_10	Liefert die prozentuale Auslastung der Automation Engine während der letzten 10 Minuten	
SYS_BUSY_60	Liefert die prozentuale Auslastung der Automation Engine während der letzten Stunde	
SYS_HOST_ALIVE	Überprüft, ob ein bestimmter Host aktiv ist	
SYS_INFO	Liest Informationen über das AE-System aus	
SYS_SERVER_ALIVE	Überprüft, ob ein bestimmter Serverprozess aktiv ist	
SYS_SNMP_ACTIVE	Prüft, ob die SNMP-Kopplung (Simple Network Management Protocol) der Automation Engine aktiv ist	&$SNMP_ACTIVE#
SYS_USER_LANGUAGE	Liefert die Sprache in der der Server die Logdateien erzeugt	&$SYS_LANGUAGE#
TOGGLE_SYSTEM_STATUS	Stoppt oder startet die automatische Verarbeitung eines gesamten Mandanten	

Datum und Zeit

[Objekte bearbeiten] [Objekte aktivieren] [Objekte lesen oder verändern] [Script-Aufbau und -Prozessierung] [Fehlerbehandlung und Meldungen] [Aktivierungsdaten] [Benutzerdaten] [Datensequenzen] [Ereignisbehandlung] [Systemzustände und -bedienung] [Datum und Zeit] [Rechnen] [Strings]

Script-Sprachmittel	Beschreibung	Vordefinierte Variable
ADD_DAYS	Addiert Tage zu einem angegebenen Datum	
ADD_PERIOD	Addiert eine Periode zu einem angegebenen Datum	
ADD_TIME	Addiert zwei Zeiten	
ADD_TIMESTAMP	Addiert Zeit zu einem Zeitstempel	
CALE_LOOK_AHEAD	Ermittelt das nächste Datum aufgrund von Kalenderbedingungen.	

CONV_DATE	Konvertiert ein Datum von einem Datumsformat in ein anderes	
CONV_TIMESTAMP	Rechnet Datum und Uhrzeit für eine andere Zeitzone um	
DAY_OF_YEAR	Liefert den laufenden Tag im Jahr	
DIFF_DATE	Ermittelt die Differenz zwischen zwei Datumsangaben in Tagen	
DIFF_TIME	Ermittelt die Differenz zwischen zwei Zeitangaben	
FIRST_OF_PERIOD	Ermittelt den ersten Tag der Periode für ein angegebenes Datum	
LAST_OF_PERIOD	Ermittelt den letzten Tag der Periode für ein angegebenes Datum	
SUB_DAYS	Subtrahiert Tage von einem angegebenen Datum	
SUB_PERIOD	Subtrahiert eine Periode von einem angegebenen Datum	
SUB_TIME	Subtrahiert zwei Zeiten	
SUB_TIMESTAMP	Subtrahiert Zeit von einem Zeitstempel	
SYS_DATE	Ermittelt das aktuelle Tagesdatum zu Beginn der Script-Prozessierung	&$DATE_ *format#*
SYS_DATE_PHYSICAL	Ermittelt das jeweils aktuelle Tagesdatum	&$PHYS_ DATE_ *format#*
SYS_LDATE	Ermittelt das logische Datum	&$LDATE_ *format#*
SYS_TIME	Stellt die aktuelle Tageszeit zu Beginn der Script-Prozessierung fest	&$TIME_ *format#*
SYS_TIME_PHYSICAL	Stellt die jeweils aktuelle Tageszeit fest	&$PHYS_ TIME_ *format#*
SYS_TIMESTAMP_ PHYSICAL	Liefert aktuelles Datum und Uhrzeit	
VALID_CALE	Prüft, ob ein Datum in einem Kalenderbegriff enthalten ist	
VALID_DATE	Prüft, ob ein Datum gültig ist	
VALID_TIME	Prüft, ob eine Zeit gültig ist	
WEEK_NR	Liefert die Kalenderwoche, in der sich ein Datum befindet	
WEEKDAY_NR	Liefern den Wochentag eines Datums als Ziffer	
WEEKDAY_XX	Gibt den Wochentag als Kürzel zurück, der für ein Datum gilt	
YEAR_9999	Extrahiert das Jahr aus einem angegebenen Datum	

Rechnen

[Objekte bearbeiten] [Objekte aktivieren] [Objekte lesen oder verändern] [Script-Aufbau und -Prozessierung] [Fehlerbehandlung und Meldungen] [Aktivierungsdaten] [Benutzerdaten] [Datensequenzen] [Ereignisbehandlung] [Systemzustände und -bedienung] [Datum und Zeit] [Rechnen] [Strings]

Script-Sprachmittel	Beschreibung
ADD	Führt eine Addition aus
DIV	Führt eine Division aus
GET_BIT	Prüft, ob in einem Bit-Feld ein bestimmtes Bit gesetzt ist
MOD	Liefert den Restwert einer Division
MULT	Führt eine Multiplikation aus
RANDOM	Generiert Zufallszahlen
SUB	Führt eine Subtraktion aus

Strings

[Objekte bearbeiten] [Objekte aktivieren] [Objekte lesen oder verändern] [Script-Aufbau und -Prozessierung] [Fehlerbehandlung und Meldungen] [Aktivierungsdaten] [Benutzerdaten] [Datensequenzen] [Ereignisbehandlung] [Systemzustände und -bedienung] [Datum und Zeit] [Rechnen] [Strings]

Script-Sprachmittel	Beschreibung
ALPHA2RUNNR	Konvertiert den Namen einer Job bzw. Reportdatei in eine RunID
ARRAY_2_STRING	Wandelt ein Script-Array in eine Zeichenkette um.
CONV_LC oder STR_LC	Setzen alle Großbuchstaben einer Zeichenkette auf Kleinbuchstaben um
CONV_UC oder STR_UC	Setzen alle Kleinbuchstaben einer Zeichenkette auf Großbuchstaben um
CONVERT	Wandelt den Datentyp eines Wertes um
FORMAT	Ändert die Formatierung einer Zahl
HEX	Konvertiert eine Zeichenkette in hexadezimale Schreibweise
ISNUMERIC	Prüft, ob eine Zeichenkette numerisch ist
MID, STR_CUT oder SUBSTR	Kopieren Zeichen aus einer Zeichenkette
RUNNR2ALPHA	Konvertiert RunID in den zugehörigen Dateinamen
STR_CAT	Verbindet zwei Zeichenketten zu einer neuen Zeichenkette
STR_ENDS_WITH	Prüft ob eine Zeichenkette mit einer bestimmten anderen Zeichenkette endet.
STR_FIND	Sucht ein Zeichen oder eine Zeichenkette innerhalb einer Zeichenkette

STR_FIND_ REVERSE	Sucht ein Zeichen oder eine Zeichenkette innerhalb einer Zeichenkette. Die Suche beginnt am Ende der zu durchsuchenden Zeichenkette.
STR_ISLOWER	Prüft die Buchstaben einer bestimmten Zeichenkette auf Kleinschreibung.
STR_ISUPPER	Prüft die Buchstaben einer bestimmten Zeichenkette auf Großschreibung.
STR_LENGTH oder STR_LNG	Ermitteln die Länge einer Zeichenkette
STR_LTRIM	Entfernt Leerzeichen am Anfang einer Zeichenkette
STR_MATCH	Vergleicht zwei Zeichenketten
STR_PAD	Zeichenkette auf eine bestimmte Länge erweitern.
STR_REVERSE	Kehrt die Reihenfolge der Zeichen einer Zeichenkette um
STR_RTRIM	Entfernt Leerzeichen am Ende einer Zeichenkette
STR_SPLIT	Zeichenkette mit Hilfe eines Trennzeichens in mehrere Teile aufspalten.
STR_STARTS_ WITH	Prüft ob eine Zeichenkette mit einer bestimmten anderen Zeichenkette beginnt.
STR_SUBSTITUTE	Ersetzt Zeichen oder Zeichenkette innerhalb einer Zeichenkette
STR_SUBSTITUTE_ VAR	Ersetzt den Namen von Script-Variablen durch deren Wert.
STR_TRIM	Entfernt Leerzeichen am Anfang und Ende einer Zeichenkette
UC_CRLF	Liefert einen Zeilenumbruch

3.2 Objekte bearbeiten

3.2.1 :REGISTER_OUTPUTFILE

Script-Anweisung: Registriert eine Datei als externen Job-Output

Syntax

:REGISTER_OUTPUTFILE *Datei, Benutzer-Login*

Syntaxteil	Beschreibung/Format
Datei	Vollqualifiziert Pfad und Name der Datei, die als Job-Output registriert werden soll. Wildcards sind nicht erlaubt! Es ist immer der absolute Pfad anzugeben! Format: Script-Literal
Benutzer-Login	Benutzer-Login verwenden Erlaubte Werte: "Y" oder "N" Format: Script-Literal

Anmerkungen

⚠ Diese Script-Anweisung kann nur in der Registerkarte **Script** von UNIX und Windows-Jobs verwendet werden!

Das Registrieren von externen als Job-Output ist auch über die Registerkarte **Output** möglich. Unterschiedlich ist jedoch der Zeitpunkt der Registrierung: Die Dateien der Registerkarte **Output** werden gleich zu Beginn der Job-Durchführung registriert, unabhängig davon ob der Job die Datei anlegen konnte oder nicht. Bei Verwendung des Sprachmittels wird die angegebene Datei erst zum Zeitpunkt des Aufrufes registriert.

Die angegebene Datei muss sich auf dem Rechner des Agenten, auf dem der Job ausgeführt wird, befinden oder von dort erreichbar sein. Sinnvoll ist es nur Dateien anzugeben, die durch den Job erzeugt werden.

Nach der Jobdurchführung wird die Datei zusätzlich zum Standard-Job-Output im Report-Dialog in der Registerkarte "Verzeichnis" aufgelistet. Dort kann die Datei direkt über das UserInterface geöffnet oder gespeichert werden.

Beispiel

In folgendem Beispiel-Script eines Windows-Jobs wird die Dateiliste des Verzeichnisses C:\temp in die Datei C:\temp\test.txt geschrieben. Anschließend wird überprüft ob das Kommando erfolgreich ausgeführt werden konnte. Ist dies der Fall so wird diese Datei als Job-Output registriert. Andernfalls bricht der Job ab.

```
dir C:\temp /S >> C:\temp\test.txt
@set retcode=%errorlevel% !
@if NOT %ERRORLEVEL% == 0 goto :retcode
:REGISTER_OUTPUTFILE "C:\temp\test.txt", "N"
```

Siehe auch:

Script-Sprachmittel - Script-Aufbau und -Prozessierung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.2.2 CREATE_OBJECT

Script-Funktion: Erstellt ein Objekt (nur Kalender, Login und Variable).

Allgemeine Informationen

Mit der Script-Funktion können ausschließlich Objekte der Objekttypen "Kalender", "Login" und "Variable" (nur statisch) erstellt werden.

⚠ Definieren Sie mit der Script-Anweisung :ON_ERROR die Reaktion auf auftretende Fehler. Wie bisher können diese mit den Script-Funktionen zur Fehlerbehandlung analysiert werden. Standardmäßig wird das Script wird weiter prozessiert. Es besteht aber auch die Möglichkeit, die Prozessierung des Scripts abzubrechen.

Sie benötigen ein Schreibrecht auf die Objekttypen Kalender, Login und Variable, damit Sie diese anlegen können.

Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

Kalender

[Kalender] [Login] [Variable]

Syntax

CREATE_OBJECT*(Objekttyp, Objektname,* [Ordner], [Titel] *)*

Syntaxteil	Beschreibung/Format
Objekttyp	Kurzbezeichnung des Objekttyps. Gültiger Wert: CALE
Objektname	Name des Objektes. Format: Script-Literal oder Script-Variable
Ordner	Name des Ordners in dem das Objekt erstellt werden soll. Format: Script-Literal oder Script-Variable
Titel	Titel des Objektes. Format: Script-Literal oder Script-Variable

Rückgabewerte
"0" - Das Kalender-Objekt wurde erfolgreich angelegt. "20644" - Ein Objekt mit dem angegebenen Namen existiert bereits. "20710" - Der Objektname enthält ein oder mehrere ungültige Zeichen.

Anmerkungen

Ein Kalender wird in dem angegebenen Ordner angelegt. Wird dieser Parameter nicht angegeben oder der Ordner existiert nicht, so wird das Objekt in <Kein Ordner> gespeichert.

Beispiele

Im ersten Beispiel wird das Objekt "FIRMENKALENDER2003" in <Kein Ordner> erstellt.

```
:SET &RET# = CREATE_OBJECT
("CALE","FIRMENKALENDAR2003",,"Firmenkalender für 2003")
```

Login

[Kalender] [Login] [Variable]

Syntax

CREATE_OBJECT*(Objekttyp, Objektname, [Ordner], [Titel])*

Syntaxteil	Beschreibung/Format
Objekttyp	Kurzbezeichnung des Objekttyps. Gültiger Wert: LOGIN
Objektname	Name des Objektes. Format: Script-Literal oder Script-Variable
Ordner	Name des Ordners, in dem das Objekt erstellt werden soll. Format: Script-Literal oder Script-Variable
Titel	Titel des Objektes. Format: Script-Literal oder Script-Variable

Rückgabewerte
"0" - Das Login-Objekt wurde erfolgreich angelegt. "20644" - Ein Objekt mit dem angegebenen Namen existiert bereits. "20710" - Der Objektname enthält ein oder mehrere ungültige Zeichen.

Anmerkungen

Ein Login-Objekt wird in dem angegebenen Ordner angelegt. Wird dieser Parameter nicht angegeben oder der Ordner existiert nicht, wird das Objekt in <Kein Ordner> gespeichert.

Beispiele

Im ersten Beispiel wird das Objekt "LOGIN.MEIER" im <Kein Ordner> erstellt.

```
:SET &RET# = CREATE_OBJECT("LOGIN","LOGIN.MEIER",,"Standard Logins")
```

Im zweiten Beispiel wird ein Login-Objekt im Order "LOGIN_STD"angelegt.

```
:SET &NEU# = CREATE_OBJECT("LOGIN","LOGIN.MEIER","LOGIN_STD",)
```

Variable

[Kalender] [Login] [Variable]

Syntax

CREATE_OBJECT*(Objekttyp, Objektname, [Ordner], [Titel], [Fehlerbehandlung], [Datentyp], [Gültigkeit])*

Syntaxteil	Beschreibung/Format
Objekttyp	Kurzbezeichnung des Objekttyps. Gültiger Wert: VARA
Objektname	Name des Objektes. Format: Script-Literal oder Script-Variable

Ordner	Name des Ordners, in welchem das Objekt erstellt werden soll. Format: Script-Literal oder Script-Variable
Titel	Titel des Objektes. Format: Script-Literal oder Script-Variable
Fehlerbehandlung	Behandlung, wenn die Variable zur Laufzeit keinen Wert enthält. Erlaubte Werte: "E" oder "I" (Standardwert) "E" = Es wird eine Fehlermeldung ausgegeben. "I" = Die Variable wird entsprechend ihres Variablentyps initialisiert.
Datentyp	Variablentyp. Erlaubte Werte: "String" (oder "C"), "Number" (oder "F"), "Timestamp" (oder "T"), "Time" oder "Date"Erlaubte Werte: "C" (Standardwert), "F" oder "T" "String", "C"= Text "Number", "F" = Zahl "Timestamp", "T" = Zeitstempel "Time" = Zeit "Date" = Datum
Gültigkeit	Gültigkeitsbereich. Erlaubte Werte: "*", "FREI" (Standardwert), "HON", "JBN", "JPN", "JPS", "USN", "USS" "*" = Kein Key "FREI" = Frei gewählt "HON" = Host - je Host-Name "JBN" = Aufgabe - je Aufgabenname "JPN" = Workflow-Name - je Workflow Name "JPS" = Workflow-Session - je Workflow Aktivierung "USN" = Benutzer - je Benutzer-Name "USS" = Benutzer-Session - je Benutzersitzung

Rückgabewerte
"0" - Das Variablen-Objekt wurde erfolgreich angelegt. "20644" - Ein Objekt mit dem angegebenen Namen existiert bereits. "20710" - Der Objektname enthält ein oder mehrere ungültige Zeichen.

Anmerkungen

Mit dieser Script-Funktion können nur statische Variablen-Objekte angelegt werden. Für mehr Informationen dazu siehe die Beschreibung der Registerkarte **Attribute** von Variablen-Objekten.

Eine Variable wird in dem angegebenen Ordner angelegt. Wird dieser Parameter nicht angegeben oder der Ordner existiert nicht, wird das Objekt in <Kein Ordner> gespeichert. Geben Sie alle optionalen Parameter nicht an, so werden die Standardwerte "Text" (für den Datentyp), "Frei gewählt" (für den Gültigkeitsbereich) und "Initialwerte" (für Key nicht gefunden) verwendet.

Beispiel

Im Beispiel wird eine Variable erstellt, in der die ermittelte Anzahl von Dateien gespeichert werden kann.

```
:SET &RET# = CREATE_OBJECT
("VARA","OUTPUT.WEBHELP.VARA","VARIABLE/TEST","Anzahl Dateien der
Hilfe","I","F","FREI")
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:ON_ERROR	Legt die Reaktion auf bestimmte Fehler bzw. Meldungen von Script-Sprachmitteln fest.
REMOVE_OBJECT	Löscht ein vorhandenes Objekt.
MODIFY_OBJECT	Ändert ein vorhandenes Objekt (nur Kalender, Login und Variable).
MOVE_OBJECT	Verschiebt ein Objekt in einen Ordner.
ACTIVATE_UC_OBJECT	Aktiviert ein Objekt.

Script-Sprachmittel - Objekte bearbeiten
Script-Sprachmittel - Fehlerbehandlung und Meldungen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.2.3 EXPORT

Script-Funktion: Exportiert Objekte in eine XML-Datei.

Syntax

EXPORT*(Objekt, Datei)*

Syntaxteil	Beschreibung/Format
Objekt	Name des zu exportierenden Objektes oder mehrerer Objekte (mit Wildcardzeichen). Format: Script-Literal oder Script-Variable Es können die Wildcardzeichen "*" und "?" verwendet werden. "*" steht dabei für beliebige, "?" für genau ein Zeichen.
Datei	Name der Datei mit kompletter Pfadangabe, in welche die Objekte exportiert werden sollen. Format: Script-Literal oder Script-Variable

Rückgabewerte
"0" - Der Export war erfolgreich. "20693" - Das Objekt existiert nicht. "21723" - Die Zieldatei existiert und ist schreibgeschützt.

Anmerkungen

Mit der Script-Funktion können Objekte in eine angegebene XML-Datei exportiert werden.

⚠ Die Import-Export-Funktionalität ist nicht für Massentransporte geeignet! Verwenden Sie in solchen Fällen den Transportkoffer.

Mit der Script-Anweisung :ON_ERROR legen Sie die Reaktion auf einen fehlerhaften Export fest. Sie können ihn mit den Script-Funktionen zur Fehlerbehandlung analysieren. Das Script wird weiter prozessiert. Es besteht aber auch die Möglichkeit, die Prozessierung des Scripts abzubrechen.

Vor dem Exportvorgang wird geprüft, ob die Datei bereits existiert. Trifft dies zu, wird sie überschrieben. Ist die Datei schreibgeschützt, kann der Export nicht durchgeführt werden.

Der Export schlägt außerdem fehl, wenn der Benutzer das Recht "Lesen" (R) auf das Objekt nicht besitzt.

Nähere Informationen über den Exportvorgang werden in den Aktivierungsreport des Objektes geschrieben, das den Export aufruft.

⚠ In einer verteilten AE-Umgebung (Workprozesse laufen auf verschiedenen Rechnern) kann nicht beeinflusst werden, auf welchem Rechner der Export durchgeführt wird! Geben Sie daher unter Windows den UNC-Pfad an. Beachten Sie, dass der Server unter einem entsprechenden Domain-User laufen muss, um auf UNC-Namen zugreifen zu können. Bei Verwendung eines UNIX-Servers muss der absolute Pfad in UNIX-Notation angegeben werden. Zusätzlich muss das Dateisystem zugreifbar sein (NFS; mount-Befehl), wobei es keine Rolle spielt auf welchem Rechner sich dieses befindet. Nur so kann sichergestellt werden, dass auch die gewünschte XML-Datei verwendet wird.

Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

Beispiel

Im folgenden Beispiel werden alle Objekte, deren Namen mit "MAWI.TAG" beginnt, exportiert. Sowohl die Objektbezeichnung, als auch der Dateinamen werden der Funktion als Script-Variable übergeben.

```
:SET&OBJEKT# = "MAWI.DAY*"
:SET&DATEI#  = "\\PCAUTOMIC\UC4global\EXPORT\uc4_export.xml"
:SET&RET#    = EXPORT(&OBJEKT#,&DATEI#)
```

Beispiel für UNIX:

```
:SET &OBJEKT# = "MAWI.DAY*"
:SET &DATEI#  = "/opt/UC4/import/uc4_export.xml"
:SET &RET#    = EXPORT(&OBJEKT#,&DATEI#)
```

Siehe auch:

Script-Sprachmittel	Beschreibung
IMPORT	Importiert Objekte aus einer XML-Datei.

Script-Sprachmittel - Objekte bearbeiten
Import und Export von Objekten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.2.4 IMPORT

Script-Funktion: Importiert Objekte aus einer XML-Datei.

Syntax

IMPORT*(Datei *[,[Ordner], [Objekt-Einstellung] [,Link-Einstellung]]* ***)***

Syntaxteil	Beschreibung/Format
Datei	Name der Datei mit kompletter Pfadangabe für den Import der Objekte Format: Script-Literal oder Script-Variable
Ordner	Name des Ordners, in welchem die Objekte gespeichert werden sollen Format: Script-Literal oder Script-Variable
Objekt-Einstellung	Einstellung, wie mit bereits vorhandenen Objekten zu verfahren ist Format: Script-Literal, Script-Variable oder Zahl Erlaubte Werte: "0" (Standardwert), "1" "0" = Vorhandene Objekte werden übersprungen "1" = Vorhandene Objekte werden überschrieben
Link-Einstellung	Einstellung, wie bestehende Ordnerverknüpfungen behandelt werden sollen Format: Script-Literal, Script-Variable oder Zahl Erlaubte Werte: "0", "1" (Standardwert) "0" = Bestehende Ordnerverknüpfungen werden nicht erhalten "1" = Bestehende Ordnerverknüpfungen werden beibehalten Dieser Parameter ist nur dann relevant, wenn als Objekt-Einstellung "1" gewählt wurde.

Rückgabewerte
"0" - Der Import war erfolgreich. "20657" - Der Zielordner existiert nicht. "20692" - Die Datei existiert nicht. "21724" - Der Zugriff auf die Datei ist wegen der fehlenden Berechtigungen nicht möglich. "21729" - Die importierte XML-Datei weist nicht das AE-Format auf und kann daher nicht importiert werden. "21730" - Die importierte XML-Datei entspricht nicht dem geforderten Encoding. "21732" - Beim Import ist ein Fehler aufgetreten. Nähere Informationen zur Ursache können Sie dem Aktivierungsreport entnehmen.

Anmerkungen

Mit der Script-Funktion können Objekte aus einer entsprechend formatierten XML-Datei importiert werden.

Die Import-Export-Funktionalität ist nicht für Massentransporte geeignet! Verwenden Sie in solchen Fällen den Transportkoffer.

Objekte werden im angegebenen Ordner angelegt. Existiert der Ordner nicht oder fehlt dieser Parameter, werden die Objekte in <Kein Ordner> gespeichert.

Mit der Script-Anweisung :ON_ERROR legen Sie die Reaktion auf einen fehlerhaften Import fest. Wie bisher können Sie ihn mit den Script-Funktionen zur Fehlerbehandlung analysieren. Das Script wird weiter prozessiert. Es besteht aber auch die Möglichkeit, die Prozessierung des Scripts abzubrechen.

Die Größe der XML-Datei ist standardmäßig auf 1024 KB beschränkt. In der Variablen UC_SYSTEM_SETTINGS kann der Administrator eine andere Größe mit dem Key "MAX_IMPORT_SIZE" definieren.

Nähere Informationen über den Importvorgang werden in den Aktivierungsreport des Objektes geschrieben, das den Import aufruft.

Der Import schlägt fehl, wenn Sie auf das Objekt oder den Zielordner keine Schreibberechtigung besitzen.

Beachten Sie, dass ein Import in den Ordner "Versionsmanagement" nicht möglich ist!

In einer verteilten AE-Umgebung (Workprozesse laufen auf verschiedenen Rechnern) kann nicht beeinflusst werden, auf welchem Rechner der Import durchgeführt wird! Geben Sie daher unter Windows den UNC-Pfad an. Beachten Sie, dass der Server unter einem entsprechenden Domain-User laufen muss, um auf UNC-Namen zugreifen zu können. Bei Verwendung eines UNIX-Servers muss der absolute Pfad in UNIX-Notation angegeben werden. Zusätzlich muss das Dateisystem zugreifbar sein (NFS; mount-Befehl), wobei es keine Rolle spielt auf welchem Rechner sich dieses befindet. Nur so kann sichergestellt werden, dass auch die gewünschte XML-Datei verwendet wird.

Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

Standardmäßig wird für jedes durch den Import überschriebene Objekt, welches sich nicht im angegebenen *Ordner* befindet, eine Verknüpfung im vorigen Ordner angelegt. Dieses Verhalten kann mit dem Parameter *Link-Einstellung* (Wert: 0) deaktiviert werden. Dadurch werden jedoch alle existierenden Verknüpfungen der importierten Objekte gelöscht!

Beispiel

Im folgenden Beispiel werden eventuell vorhandene Objekte beim Importieren übersprungen. Sowohl die Datei als auch der Ordnernamen werden der Funktion als Script-Variablen übergeben.

```
:SET&DATEI#   = "\\PCAUTOMIC\UC4global\IMPORT\uc4_import.xml"
:SET&ORDNER# = "IMPORT/JOBS"
:SET&RET#     = IMPORT(&DATEI#,&ORDNER#,"0")
```

Beispiel für UNIX:

```
:SET  &DATEI#   = "/opt/UC4/import/uc4_import.xml"
:SET  &ORDNER# = "IMPORT/JOBS"
:SET  &RET#     = IMPORT(&DATEI#,&ORDNER#,"0")
```

Siehe auch:

Script-Sprachmittel	Beschreibung
EXPORT	Exportiert Objekte in eine XML-Datei.

Script-Sprachmittel - Objekte bearbeiten

Import und Export von Objekten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.2.5 MODIFY_OBJECT

Script-Funktion: Ändert ein vorhandenes Objekt (nur Kalender, Login und Variable).

Allgemeine Informationen

Mit der Script-Funktion können ausschließlich Objekte der Objekttypen "Kalender", "Login" und "Variable" (nur statisch) geändert werden.

Definieren Sie mit der Script-Anweisung :ON_ERROR die Reaktion auf auftretende Fehler. Wie bisher können diese mit den Script-Funktionen zur Fehlerbehandlung analysieren. Standardmäßig wird das Script wird weiter prozessiert. Es besteht aber auch die Möglichkeit, die Prozessierung des Scripts abzubrechen.

Sie benötigen ein Schreibrecht auf die Objekttypen Kalender, Login und Variable, damit Sie diese anlegen können.

Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

Kalender

[Kalender] [Login] [Variable]

Syntax

MODIFY_OBJECT*(Objektname, [Titel], [Kalenderbegriff] [,[Datumsformat:]Datum1] [, [Datumsformat:]Datum2]])*

Syntaxteil	Beschreibung/Format
Objektname	Name des Kalender-Objektes. Format: Script-Literal oder Script-Variable
Titel	Titel des Kalender-Objektes. Format: Script-Literal oder Script-Variable
Kalenderbegriff	Name eines Kalenderbegriffs des angegebenen Kalender-Objekts. Format: Script-Literal oder Script-Variable
Datumsformat	Formatvorgabe für das angegebene Datum.
: oder ;	Trennzeichen zwischen dem *Datumsformat* und Datum.
Datum1	Anfangsdatum des Gültigkeitszeitraums vom Kalenderbegriff. Angabe eines Datums, welches dem *Datumsformat* entspricht.
Datum2	Endedatum des Gültigkeitszeitraums vom Kalenderbegriff. Angabe eines Datums, welches dem *Datumsformat* entspricht.
[Datumsformat:]Datum1 und [Datumsformat:]Datum2	Beginn und Ende der Gültigkeit des Kalenderbegriffes. Format: Script-Literal oder Script-Variable

Rückgabewerte
"0" - Das Kalender-Objekt wurde erfolgreich geändert. "20645" - Das Objekt existiert nicht. "20670" - Das Endedatum für den Kalenderbegriff liegt vor dem Anfangsdatum.

Anmerkungen

Mit der Script-Funktion können Sie den Titel des Kalender-Objektes, sowie das Anfangs- und Endedatum eines Kalenderbegriffs ändern. Nicht verwendete Parameter lassen bestehende Kalenderdefinitionen unverändert.

Für die Behandlung von Kalenderbegriffen existieren noch folgende Script-Sprachmittel:

:SET_CALE - Fügt einem Kalenderbegriff ein Datum oder einen Zeitraum hinzu oder entfernt diese.
VALID_CALE - Prüft, ob ein Datum in einem Kalenderbegriff enthalten ist.

Beispiel

Im Beispiel wird der Gültigkeitszeitraum des Kalenderbegriffes "ARBEITSTAGE" geändert. Der Titel des Kalenders bleibt unverändert.

```
:SET  &RET# = MODIFY_OBJECT
(
"FIRMENKALENDER"
,,"ARBEITSTAGE","DD.MM.YYYY:01.05.2011","DD.MM.YYYY:01.05.2012")
```

Login

[Kalender] [Login] [Variable]

Syntax

MODIFY_OBJECT*(Objektname, [Titel], Name, Typ, Login Info, [Passwort], [Aktion])*

Syntaxteil	Beschreibung/Format
Objektname	Name des Login-Objekts. Format: Script-Literal oder Script-Variable
Titel	Titel des Login-Objekts. Format: Script-Literal oder Script-Variable
Name	Name eines Agenten bzw. Backend-Systems. Format: Script-Literal oder Script-Variable Geben Sie "*" an, so gilt der Login-Eintrag für alle Agenten bzw. Systeme. Der Name von Backend-Systemen wird im Login-Objekt vom Benutzer definiert.

Typ	Hosttyp bzw. Applikation. Format: Script-Literal oder Script-Variable Erlaubte Werte: "BS2000" = Host vom Typ BS2000/OSD "GCOS8" = Host vom Typ Bull GCOS 8 "JMX" = Login-Eintrag für Java EE/JMX "MAIL" = Login-Eintrag die Mail-Schnittstelle "MPE" = Host vom Typ MPE "MVS" = Host vom Typ z/OS, MVS, z/OS "OA" = Login-Eintrag für Oracle Applications "OS400" = Host vom Typ OS/400 "PS" = Login-Eintrag für PeopleTools "R3" = Login-Eintrag für SAP "SIEBEL" = Login-Eintrag für Siebel "SQL" = Login-Eintrag für Datenbanken "UNIX" = Host vom Typ Unix, Linux, z/Linux "VMS" = Host vom Typ OpenVMS "WINDOWS" = Host vom Typ Windows Neben den aufgelisteten Plattformen der Agenten können auch alle Typen angegeben werden, die vom Administrator in der Variable UC_LOGIN_TYPES definiert wurden.
Login Info	Login-Informationen, mit denen eine Anmeldung erfolgen soll. Format: Script-Literal oder Script-Variable Das Format der Login Info ist Plattform- und Anwendungsspezifisch. Details dazu finden Sie im Kapitel "Login-Objekt".
Passwort	Passwort für die Plattform der Anwendung. Format: Script-Literal oder Script-Variable Es erfolgt keine Plausibilitätsprüfung beim Ausführen der Script-Funktion MODIFY_OBJECT.
Aktion	Aktion die ausgeführt werden soll. Erlaubte Werte: "ADD" (Standardwert), "DEL" "ADD" = Fügt den angegebenen Benutzereintrag am Ende der Liste hinzu, bzw. überschreibt einen bestehenden, mit identischen Daten für Host, Hosttyp und Login Info. "DEL" = Entfernt den angegebenen Benutzereintrag aus der Liste. Ist der Benutzereintrag in der Liste nicht vorhanden, wird dennoch kein Fehlercode erzeugt. Die Angabe eines Passwortes ist beim Löschen des Eintrages nicht relevant.

Rückgabewerte
"0" - Das Login-Objekt wurde erfolgreich geändert. "20645" - Das Objekt existiert nicht.

Anmerkungen

Mit der Script-Funktion können Sie den Titel, sowie die Login-Einträge eines Login-Objektes ändern.

Die Verwendung der Funktion MODIFY_OBJECT für das Login-Objekt dient vorwiegend zur automatisierten Benutzerverwaltung um z.B. extern Passwortänderungen in der Automation Engine durchzuführen.

Wird ein Agent inklusive Typ angegeben, der im Login-Objekt bereits vorhanden ist, so wird der bestehende Eintrag geändert. Stimmt Agent und Typ dabei nicht überein, so hat die Scriptzeile keine Auswirkung.

Beispiel

Im ersten Beispiel werden im Login-Objekt "LOGIN.MEIER" die Anmeldedaten für den Benutzer "meier" auf dem Host "UNIX01" gesetzt und dabei das Passwort "uc4" vergeben. Existiert der Benutzereintrag bereits, wird dabei nur das Passwort auf "uc4" gesetzt.

```
:SET  &RET# = MODIFY_OBJECT
("LOGIN.MEIER",,"UNIX01","UNIX","meier","uc4","ADD")
```

Im folgenden Beispiel werden im Login-Objekt "LOGIN.MEIER" die Anmeldedaten für den Benutzer "meier" im Mandant "012" des SAP-Systems "SAP01" gelöscht.

```
:SET  &RET# = MODIFY_OBJECT
("LOGIN.MEIER",,"SAP01","R3","012,meier","","DEL")
```

Variable

[Kalender] [Login] [Variable]

Syntax

MODIFY_OBJECT(*Objektname*, *[Titel]*, *[Fehlerbehandlung]*, *[Datentyp]*)

Syntaxteil	Beschreibung/Format
Objektname	Name des Variablen-Objekts. Format: Script-Literal oder Script-Variable
Titel	Titel des Variablen-Objekts. Format: Script-Literal oder Script-Variable
Fehlerbehandlung	Behandlung wenn die Variable zur Laufzeit keinen Wert enthält. Erlaubte Werte: "E" oder "I" "E" = Es wird eine Fehlermeldung ausgegeben. "I" = Die Variable wird entsprechend ihres Variablentypes initialisiert.
Datentyp	Variablentyp. Erlaubte Werte: "String" (oder "C"), "Number" (oder "F"), "Timestamp" (oder "T"), "Time" oder "Date" "String", "C"= Text "Number", "F" = Zahl "Timestamp", "T" = Zeitstempel "Time" = Zeit "Date" = Datum

Rückgabewerte
"0" - Das Variable-Objekt wurde erfolgreich geändert. "20640" - Beim Datentyp wurde ein ungültiger Wert angegeben. "20645" - Das Objekt existiert nicht. "20651" - Der Datentyp kann nicht geändert werden, da sich Werte in der Variable befinden.

Anmerkungen

Mit der Script-Funktion können Sie den Titel, sowie die Fehlerbehandlung und den Datentyp eines statischen Variable-Objektes ändern.

Soll der Datentyp geändert werden, dürfen keine Werte in der Variablen gespeichert sein.

Dynamische Variablen-Objekte können mit der Script-Funktion nicht geändert werden.

Beispiel

Im Beispiel wird eine Variable geändert, so dass darin die ermittelte Anzahl von Dateien gespeichert werden kann.

```
:SET  &RET# = MODIFY_OBJECT("OUTPUT.WEBHELP.VARA","Anzahl Dateien der
Hilfe mit Bildern",,"F")
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:ON_ERROR	Legt die Reaktion auf bestimmte Fehler bzw. Meldungen von Script-Sprachmitteln fest.
CREATE_OBJECT	Erstellt ein Objekt (nur Kalender, Login und Variable).
REMOVE_OBJECT	Löscht ein vorhandenes Objekt.
MOVE_OBJECT	Verschiebt ein Objekt in einen Ordner.
ACTIVATE_UC_OBJECT	Aktiviert ein Objekt.

Script-Sprachmittel - Objekte bearbeiten
Script-Sprachmittel - Fehlerbehandlung und Meldungen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.2.6 MOVE_OBJECT

Script-Funktion: Verschiebt ein Objekt in einen Ordner.

Syntax

MOVE_OBJECT ***(Objektname, Zielordner)***

Syntaxteil	Beschreibung/Format
Objektname	Name des Objektes. Format: Script-Literal oder Script-Variable
Zielordner	Pfad des Ordners, in den das Objekt verschoben werden soll. Format: Script-Literal oder Script-Variable

Rückgabewerte
"0" - Das Objekt wurde erfolgreich verschoben. "20645" - Das Objekt existiert nicht. "20657" - Der Zielordner existiert nicht.

Anmerkungen

In welchem Ordner sich das zu verschiebende Objekt befindet spielt keine Rolle, da dieser automatisch ermittelt wird. Besitzt das Objekt Verknüpfungen, so werden diese ignoriert und nicht verschoben.

Es kommt zu einem Laufzeitfehler, wenn Sie kein Schreibrecht auf das Objekt oder den Zielordner besitzen.

Existiert das Objekt bzw. der Ordner nicht, so kommt es nur dann zu einem Laufzeitfehler, wenn die Script-Anweisung :ON_ERROR mit dem Parameter ABEND verwendet wurde. In letzterem Fall verbleibt das Objekt in seinem ursprünglichen Ordner.

Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

Beispiel

Im Beispiel wird das Objekt "WIEN" in den Unterordner "OBJEKTE" des Ordners "ZEITZONEN" verschoben.

```
:ON_ERROR ABEND
:SET &RET# = MOVE_OBJECT ("WIEN","ZEITZONEN/OBJEKTE")
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:ON_ERROR	Legt die Reaktion auf bestimmte Fehler bzw. Meldungen von Script-Sprachmitteln fest.
CREATE_OBJECT	Erstellt ein Objekt (nur Kalender, Login und Variable).
REMOVE_OBJECT	Löscht ein vorhandenes Objekt.
MODIFY_OBJECT	Ändert ein vorhandenes Objekt (nur Kalender, Login und Variable).

Script-Sprachmittel - Objekte bearbeiten
Script-Sprachmittel - Fehlerbehandlung und Meldungen

Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.2.7 REMOVE_OBJECT

Script-Funktion: Löscht ein vorhandenes Objekt.

Syntax

REMOVE_OBJECT ***(Objektname)***

Syntaxteil	Beschreibung/Format
Objektname	Name des zu löschenden Objektes. Format: Script-Literal oder Script-Variable

Rückgabewerte
"0" - Der Löschvorgang war erfolgreich. "20645" - Das Objekt existiert nicht. "20217" - Das Objekt ist gerade zur Bearbeitung geöffnet. "20369" - Das Objekt befindet sich im Transportkoffer.

Anmerkung

Ein gelöschtes Objekt wird in den Papierkorb verschoben und kann daraus wiederhergestellt werden.

Mit der Script-Anweisung :ON_ERROR legen Sie die Reaktion auf einen Fehler fest und können diesen mit den Script-Funktionen zur Fehlerbehandlung analysieren. Das Script wird je nach Einstellung entweder weiter prozessiert oder abgebrochen.

Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

Beispiel

Im Beispiel soll das Objekt "FIRMENKALENDER2003" gelöscht werden.

```
:SET &RET# = REMOVE_OBJECT("FIRMENKALENDER2003")
```

Siehe auch:

Script-Sprachmittel	Beschreibung
CREATE_OBJECT	Erstellt ein Objekt (nur Kalender, Login und Variable).

MODIFY_ OBJECT	Ändert ein vorhandenes Objekt (nur Kalender, Login und Variable).
MOVE_OBJECT	Verschiebt ein Objekt in einen Ordner.
:ON_ERROR	Legt die Reaktion auf bestimmte Fehler bzw. Meldungen von Script-Sprachmitteln fest.

Script-Sprachmittel - Objekte bearbeiten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.3 Objekte aktivieren

3.3.1 :BEGINREAD... :ENDREAD

Script-Anweisungen: Beginn und Ende eines Dialogfeldes für Benutzerabfragen.

Syntax

:BEGINREAD*[Titel[,***F***]]*
:ENDREAD

Syntaxteil	Beschreibung/Format
:BEGINREAD	Beginn des Dialogfeldes für Benutzerabfragen.
Titel	Titel des Dialogfeldes. Format: Script-Literal, Script-Variable oder Script-Funktion Standardwert: Name des Objektes
F	Formatiert den Text des Dialogfeldes so, dass alle Buchstaben dieselbe Breite aufweisen.
:ENDREAD	Ende des Dialogfeldes für Benutzerabfragen.

Anmerkungen

Die Script-Anweisungen :BEGINREAD und :ENDREAD definieren den Beginn und das Ende eines Dialogfeldes für Benutzerabfragen. Zwischen beiden Script-Anweisungen können beliebige :READ-Anweisungen stehen, deren Anzahl die Größe des Dialogfeldes bestimmt. Die Parameter sind ausschlaggebend für das Aussehen des Dialogfeldes und die Funktion der erzeugten Eingabefelder, die die Benutzereingaben entgegennehmen.

Die Angabe eines Titels für das Dialogfeld ist optional. Wenn Sie *Titel* nicht verwenden, wird der Name des Objektes angezeigt. Die Angabe des Parameters "F" bewirkt, dass alle Buchstaben dieselbe Breite aufweisen. Dadurch ist eine spaltengenaue Ausrichtung des Textes möglich.

Verwenden Sie die :PRINT-Anweisung, um beliebigen Text innerhalb des Dialogfeldes auszugeben.

Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

Zur Laufzeit generieren

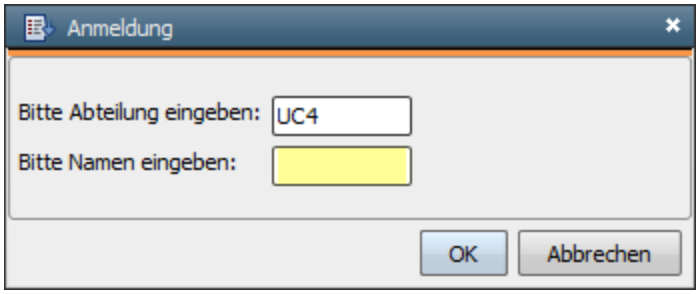
Das Dialogfeld wird nicht angezeigt, wenn die Option "Zur Laufzeit generieren" (Registerkarte **Attribute**) aktiviert ist. Dasselbe gilt auch, wenn das Objekt in einem Workflow läuft, das diese Einstellung besitzt.

Für die :READ-Anweisungen werden die Vorgabewerte verwendet. Wenn dadurch keine gültige Antwort entsteht, wird die Generierung des Scripts abgebrochen.

Beispiel

Das Beispiel gibt das Dialogfeld "Anmeldung" aus, in dem der Benutzer zur Eingabe der Abteilung und des Namens aufgefordert wird.

```
:BEGINREAD  "Anmeldung"
:READ  &ABT#, "08", "Bitte Abteilung eingeben",, M
:READ  &NAME#, "08", "Bitte Namen eingeben",, M
:ENDREAD
```



Siehe auch:

:READ	Fragt Angabe vom Benutzer im Dialog ab.
:PRINT	Dient zur Ausgabe von Text in einem Dialogfeld für Benutzerabfragen oder im Aktivierungsprotokoll eines Objektes.

Script-Sprachmittel - Objekte aktivieren
Beispielsammlung:
Datenbankwartung mit Optionen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.3.2 :PRINT

Script-Anweisung: Dient zur Ausgabe von Text in einem Dialogfeld für Benutzerabfragen oder im Aktivierungsprotokoll eines Objektes.

Syntax

:P[RINT] ***Text***
:P[RINT] ***Text1, Text2***

Syntaxteil	Beschreibung/Format
Text, Text1, Text2	Text, der im Dialogfeld oder im Aktivierungsprotokoll ausgegeben werden soll, maximal 1024 Zeichen für *Text* oder *Text1* + *Text2*. Format: Script-Literal oder Script-Variable

Anmerkungen

Für die Script-Anweisung gibt es zwei Einsatzmöglichkeiten:

Anzeige im Aktivierungsprotokoll

:PRINT gibt den angegebenen Text im Aktivierungsprotokoll aus. Dies kann dazu genutzt werden, um Verarbeitungsergebnisse anzuzeigen. Auch ist es damit möglich, die korrekte Ersetzung von Script-Variablen zu überprüfen. Das Aktivierungsprotokoll ist Teil des Reports.

Syntax	Beschreibung
:PRINT *Text*	Der Text wird in einer eigenen Zeile angezeigt.
:PRINT *Text1*, *Text2*	Für Text1 und Text2 wird jeweils eine eigene Zeile angelegt.

⚠ Ein Zeilenumbruch wird nur dann eingefügt, wenn Text1 mit einfachen oder doppelten Hochkommata angegeben wird. Für Text2 spielt dies keine Rolle.

Anzeige im Dialogfeld

:PRINT gibt den angegebenen Text in Dialogfelder für Benutzerabfragen (:BEGINREAD... :ENDREAD) aus. Nutzen Sie dies, um die Eingabemaske übersichtlicher zu gestalten und Erklärungen hinzuzufügen.

Syntax	Beschreibung
:PRINT *Text*	Der Text wird in einer eigenen Zeile angezeigt.
:PRINT *Text1*, *Text2*	Folgt nach der Anweisung ein :READ so wird Text1 über der Eingabeaufforderung und Text2 über dem Textfeld positioniert. Ansonsten wird Text2 an Text1 angehängt.

ℹ Für beide Einsatzmöglichkeiten gilt, dass die in den Texten enthaltenen Script-Variablen durch ihre Werte ersetzt werden.

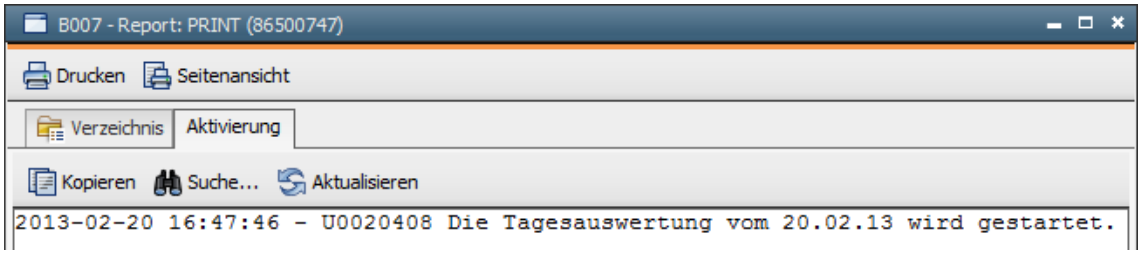
Der Administrator kann mit der Einstellung SERVER_OPTIONS in der Variablen UC_SYSTEM_SETTINGS bestimmen, ob die Ausgabe der Script-Anweisung zusätzlich zum Aktivierungsprotokoll auch in die Log-Datei / Report des Automation Engine geschrieben werden soll.

Beachten Sie, dass :PRINT die Leerzeichen am Ende des auszugebenden Textes abschneidet!

Beispiele

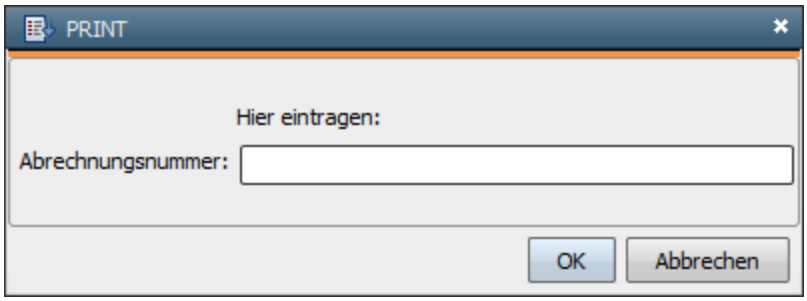
Dieses Beispiel zeigt die Ausgabe eines Textes im Aktivierungsprotokoll, in dem gleichzeitig das aktuelle Tagesdatum in einer Script-Variablen verwendet wird.

```
:SET&TAGESDATUM# = SYS_DATE(DD.MM.YY)
:PRINT"Die Tagesauswertung vom &TAGESDATUM# wird gestartet."
```



Im Beispiel wird der Text "Hier eintragen:" im Dialogfeld für Benutzerabfragen über dem Eingabefeld ausgegeben.

```
:BEGINREAD
:PRINT"", "Hier eintragen:"
:READ&ANR#,,"Abrechnungsnummer"
:ENDREAD
```



Siehe auch:

:BEGINREAD... :ENDREAD	Beginn und Ende eines Dialogfeldes für Benutzerabfragen.
:READ	Fragt Angabe vom Benutzer im Dialog ab.

Script-Sprachmittel - Objekte aktivieren
Beispielsammlung:
Datenbankwartung mit Optionen
Endestatus aufgrund des Reportinhaltes setzen
Ermittlung der Fehlermeldung und -nummer
Reaktion auf externe Ereignisse
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.3.3 :PUT_READ_BUFFER, :PUT_PROMPT_BUFFER

Script-Anweisung: Stellt Namen und Inhalt einer Script-Variablen in den Zwischenspeicher.

Syntax

:PUT_READ_BUF[FER] ***Variablenname = Inhalt***
:PUT_PROMPT_BUF[FER] ***Variablenname = Inhalt***

Syntaxteil	Beschreibung/Format
Variablenname	Name der Script-Variable (ohne &). Format: AE-name
Inhalt	Wert, welcher der Script-Variablen zugewiesen wird. Format: Script-Literal oder Script-Variable

Anmerkungen

Die Script-Anweisung ist in Verbindung mit der Funktion ACTIVATE_UC_OBJECT wie folgt zu verwenden:

1. Script-Variablen mit :PUT_READ_BUFFER speichern.
2. Mit ACTIVATE_UC_OBJECT ein Objekt aktivieren.
3. Nun kann im Script des aktivierten Objektes mittels :READ auf die gespeicherten Script-Variablen zugegriffen werden.

:PUT_READ_BUFFER legt die Script-Variable in einem eigenen Speicherbereich ab und vergibt für diesen eine interne ID. Rufen Sie für jeden Wert, den Sie speichern wollen, diese Anweisung auf. Er wird in demselben Speicherbereich abgelegt.

Der Parameter *Variablenname* wird ohne das bei Script-Variablen übliche Zeichen "&" angegeben.

Gesamte Script-Arrays können nicht in den Zwischenspeicher geschrieben werden!

Sobald nun ACTIVATE_UC_OBJECT im Script aufgerufen wird, gilt der Speicherbereich als abgeschlossen. Das aktivierte Objekt kann auf dessen Script-Variablen, wie zuvor erwähnt, zugreifen.

Die :PUT_READ_BUFFER-Anweisung, die nach der Funktion ACTIVATE_UC_OBJECT steht, benutzt einen anderen Speicherbereich und vergibt ihm auch eine neue ID.

Beim Auslesen der Script-Variablen aus dem Zwischenspeicher gilt Folgendes:

- Mit der Anweisung :READ werden die Script-Variablen einzeln ausgelesen. Dabei müssen Sie jedoch das Zeichen "&" vor den Variablennamen stellen.
- Jede Script-Variable kann nur einmal ausgelesen werden, da sie beim Lesevorgang im Zwischenspeicher gelöscht wird.

Das Sprachmittel dient ebenfalls dazu, die PromptSet-Werte für das mit ACTIVATE_UC_OBJECT aktivierte Objekt zu setzen. Der Name der ReadBuffer-Variable muss dabei dem Namen einer PromptSet-Variable des gestarteten Objekts entsprechen.

Mehrfache Verwendung von Variablennamen

Es ist auch möglich :PUT_READ_BUFFER mehrmals mit demselben Variablennamen aufzurufen. Der Wert wird dabei nicht überschrieben, da die Anweisung für jeden einzelnen Aufruf einen eigenen Eintrag im Speicherbereich anlegt. Beim Auslesen einer Script-Variablen beginnt :READ immer am Anfang des Speicherbereiches zu suchen. Der Wert des ersten Vorkommens der Variablen wird zurückgeliefert. Anschließend erfolgt die Löschung des Eintrages. Deshalb wird beim nächsten Auslesen der Script-Variablen deren nächstes Vorkommen zurückgeliefert. Dies bietet einen immensen Vorteil, da Sie die Werte in einer Schleife setzen und auslesen können.

Beispiele

Das erste Beispiel versorgt die Script-Variable "DATUM1" mit einem Wert und stellt den Eintrag in den Zwischenspeicher. Danach wird der Job "AUSWERTUNG" aktiviert.

```
:PUT_READ_BUFFER DATUM1# = "01.01.2006"
:SET &AKTJOB# = ACTIVATE_UC_OBJECT( AUSWERTUNG)
```

Der Job "AUSWERTUNG" verwendet diesen Eintrag im Zwischenspeicher.

```
:READ &DATUM1#,,
:SET &RET# = VALID_DATE("TT.MM.JJJJ:&DATUM1#")
```

Im zweiten Beispiel wird ein bestimmter Wert eines Script-Arrays in den Zwischenspeicher geschrieben. Beachten Sie, dass es nicht möglich ist, ein komplettes Array nicht in den Zwischenspeicher zu stellen.

```
:DEFINE &ARRAY#,string,5
:SET &ARRAY#[1] = "test1"
:PUT_READ_BUFFER TEST# = &ARRAY#[1]
:SET &AKTJOB# = ACTIVATE_UC_OBJECT(PRINT)
```

Das aktivierte Objekt "PRINT" liest den Wert aus dem Zwischenspeicher und schreibt diesen in das Aktivierungsprotokoll.

```
:READ &TEST#,,
:PRINT "&TEST#"
```

Das dritte Beispiel schreibt mehrere Werte für ein Prompt in den Prompt-Buffer, wenn das Kästchen für die Mehrfachauswahl angehakt ist. In diesem Fall ist "VARA#" der Name der Variable, die als Datenreferenz für das PromptSet-Element ausgewählt wurde.

```
:PUT_PROMPT_BUFFER"VARA#[]"="VALUE1"
:PUT_PROMPT_BUFFER"VARA#[]"="VALUE2"
```

Siehe auch:

Sprachmittel	Beschreibung
:READ	Fragt Angabe vom Benutzer im Dialog ab.
ACTIVATE_UC_OBJECT	Aktiviert ein Objekt.

Script-Sprachmittel - Objekte aktivieren
Beispielsammlung
Benachrichtigung mit variablem Meldungstext
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.3.4 :READ

Script-Anweisung: Fragt Angabe vom Benutzer im Dialog ab.

Syntax

:REA[D] ***InVariable,*** *[Eingabeprüfung], [Eingabeaufforderung] [, Vorgabewert] [, Eingabeformatierung]*

Syntaxteil	Beschreibung/Format
InVariable	Nimmt den vom Benutzer eingegebenen Wert auf. Format: Script-Variable
Eingabeprüfung	Kennzeichen, welches festlegt, wie die Benutzereingabe zu überprüfen ist. Format: Script-Literal oder Script-Variable Folgende Kennzeichen können gesetzt werden: • "00" Es erfolgt keine Eingabeprüfung (Standardwert). • "01" bis "99" Legt die maximale Länge der eintragbaren Zeichen fest (z.B. "10"). • "*min. Zahl - max. Zahl*" Die eingegebene Zahl muss in diesem Bereich liegen (z.B."10-20"). • Datumsformat Die Benutzereingabe muss mit dem vorgegebenen Datumsformat übereinstimmen (z.B. JJJJMMTT). • *Wert*, *Wert1 - Wert2* Die Benutzereingabe muss mit einem festgelegten Wert oder Wertebereich identisch sein. Werte und Wertebereiche können beliebig kombiniert, durch Komma getrennt, angegeben werden.
Eingabeaufforderung	Text, der den Benutzer im Dialog zur Eingabe auffordert. Format: Script-Literal oder Script-Variable Standardwert: "Bitte Wert für Variable *Variablenname* eingeben"
Vorgabewert	Text, der im Dialog als Eingabe vorgeschlagen wird. Format: Script-Literal oder Script-Variable Maximal 1024 Zeichen ⚠ Ein leerer String kann nicht als Vorgabewert verwendet werden. Ein Leerzeichen sich muss innerhalb von Anführungszeichen befinden.

Eingabeformatierung	Kennzeichen, wie die Eingabe formatiert werden soll. Es können mehrere Kennzeichen in beliebiger Reihenfolge angegeben werden (z.B. "MN"). Format: Script-Literal oder Script-Variable Es gelten folgende Kennzeichen: • "C" Der Cursor wird auf dieses Eingabefeld positioniert, wenn sich die READ-Anweisung innerhalb eines :BEGINREAD... :ENDREAD Blockes befindet. • "D" Die Eingabe erfolgt geschützt und wird mit Sternchen "*" dargestellt. • "I" (nur bei Jobs) Im Objekt "Job" werden die :READ-Anweisungen und die erfolgten Benutzereingaben standardmäßig im generierten Job als REMARK-Zeilen hinterlegt. Die Angabe des Kennzeichens "I" unterdrückt dieses Protokollieren. • "K" Die Eingabe ist in Kleinbuchstaben möglich. Ohne dieses Kennzeichen werden die Zeichen schon bei der Eingabe in Großbuchstaben umgesetzt. Die Option "K" steht nur für Eingaben im UserInterface zur Verfügung und hat keinen Einfluss auf den Read-Buffer. • "M" Der Benutzer muss etwas eingeben. • "N" Es ist nur die Eingabe numerischer Zeichen möglich. • "O" Der Benutzer kann in einem Listenfeld nicht nur dessen Einträge auswählen, sondern auch selbst etwas eintragen. Die Kennzeichen "D" und "N" können nicht gemeinsam verwendet werden. Verwenden Sie den Parameter *Eingabeprüfung* für geschützte Eingaben, die nur Zahlen entgegennehmen sollen. Beispiel: `:READ &PASS#,"1-99999999","Passwort (numerisch) eingeben",,"DM"` Wenn Sie eine *Eingabeprüfung* verwenden jedoch keinen *Vorgabewert*, ist trotzdem dessen Komma zu setzen. Sie sehen dies anhand des vorigen Beispiels.

Anmerkungen

Durch die :READ-Anweisung wird ein Dialogfeld ausgegeben, dessen Aussehen und Funktion von den verwendeten Parametern abhängt. Der Name des Objektes ist in der Titelzeile dargestellt. Über ein Eingabefeld wird die Benutzereingabe entgegengenommen und in eine Script-Variable eingelesen.

⚠ Beachten Sie, dass das Dialogfeld nur angezeigt wird, wenn das Script im Dialog-Modus läuft! Ist dies nicht der Fall, werden die Vorgabewerte verwendet.

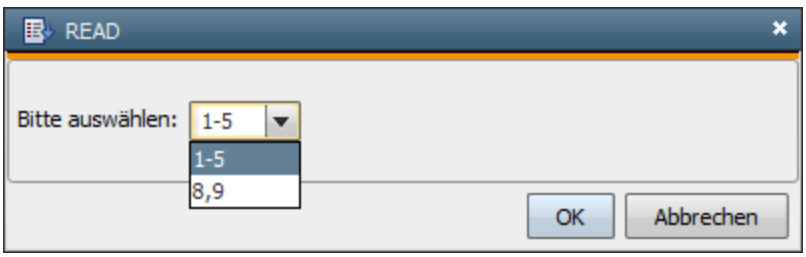
⚠ Achten Sie bei der Verwendung dieses Scriptsprachmittels, dass der eingegebene Wert mit dem Datentyp der "InVariable" (Zielvariable) kompatibel ist.

Die Parameter *Eingabeprüfung*, *Eingabeaufforderung*, *Vorgabewert* und *Eingabeformatierung* sind optional. Sollen für *Eingabeprüfung* und *Eingabeaufforderung* keine Inhalte vorgegeben werden, sind in der :READ-Anweisung dennoch die Kommata zu setzen.

Feld	Beschreibung
Zahlenfeld	Verwenden Sie in der *Eingabeprüfung* einen durchgehenden numerischen Wertebereich (z.B. "0-255") mit der Eingabeformatierung "N", so wird ein Zahlenfeld mit auf/ab Pfeilen angezeigt.
Listenfeld	Enthält der Parameter *Eingabeprüfung* eindeutige Werte, z.B.: "A,B,C", werden diese in Form eines Listenfeldes ausgegeben.
Textfeld	Werden in der *Eingabeprüfung* Wertebereiche verwendet, beispielsweise "A, 5-9", wird ein Textfeld erzeugt.

Sollen Komma und Bindestrich nicht als Trennkennzeichen interpretiert werden, können die Werte zusätzlich unter einfache Hochkomma gesetzt werden. Beispiel: Es kann entweder der Wert "1-5" oder der Wert "8,9" ausgewählt werden.

```
:READ &OPTION#,"'1-5','8,9'","Bitte auswählen","1-5"
```



Über die Eingabeprüfung ist es möglich nur Buchstaben, nur Zahlen oder auch ein Gemisch aus beidem zuzulassen. Bei letzterem erfolgt die Kontrolle über die Zeichen. Daher muss in solchen Fällen die Eingabeprüfung wie folgt formuliert werden. Das Beispiel lässt Eingaben von "A" bis "F" und "1" bis "999" zu:

```
:READ &EINGABE#, "A-F,1-9,01-99,001-999","Bitte auswählen"
```

⚠ Wenn der Benutzer keinen Wert einträgt oder auswählt, wird in der Script-Variablen ein Leerzeichen " " gespeichert!

ℹ Im Report wird automatisch mitprotokolliert welche Werte in den Dialogfeldern eingegeben wurden. Im Jobreport können Sie dies mit dem Parameter "I" deaktivieren. Beachten Sie, dass geschützte Eingaben (in der Regel Passwörter), die mit dem Parameter "D" definiert werden, im Report verschlüsselt dargestellt sind.

ℹ Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

ℹ Mit der :READ-Anweisung ist es ebenfalls möglich Werte, die zuvor in den Zwischenspeicher gestellt wurden, auszulesen (siehe :PUT_READ_BUFFER).

⚠ Bei der Verwendung von :PUT_VAR vor einer :READ-Anweisung gilt folgende Besonderheit. Wird bei :READ die Generierung des Scripts manuell über die Schaltfläche "Abbrechen" oder aufgrund ungültiger Vorgabewerte (siehe "Zur Laufzeit generieren") abgebrochen, so besitzt das Variablen-Objekt trotzdem die mit :PUT_VAR gesetzten Werte.

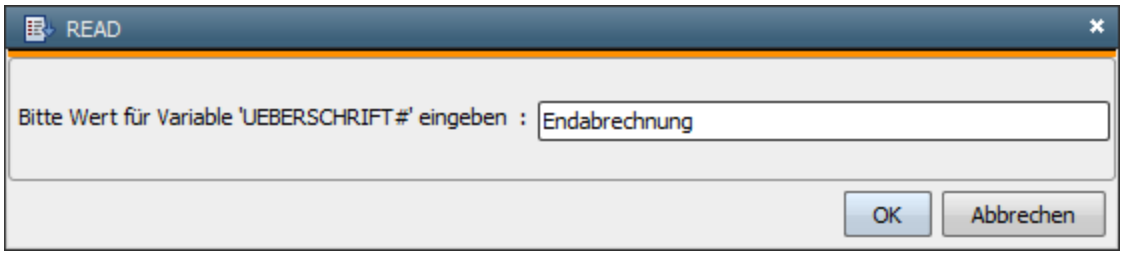
Beispiele

Abfrage einer Überschrift vom Benutzer. Der eingelesene Wert wird keiner Prüfung unterzogen. Auch eine Leereingabe ist möglich. Die Zeichen werden schon bei der Eingabe auf Großbuchstaben umgesetzt.

```
:READ &UEBERSCHRIFT#,"00","Bitte Überschrift eingeben"
```

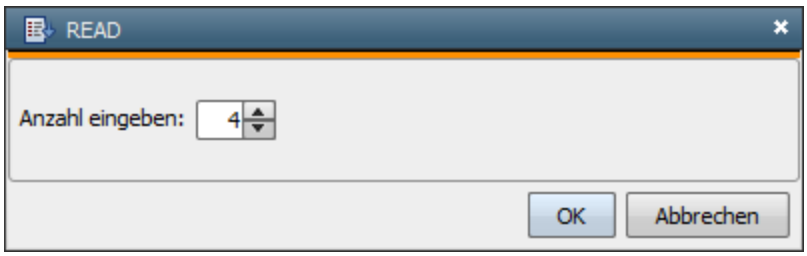
Abfrage einer Überschrift vom Benutzer. Es erfolgen keinerlei Vorgaben. Auch hier wird der eingegebene Wert nicht geprüft. Die Eingabe von Kleinbuchstaben ist jedoch möglich.

```
:READ &UEBERSCHRIFT#,,,,"K"
```



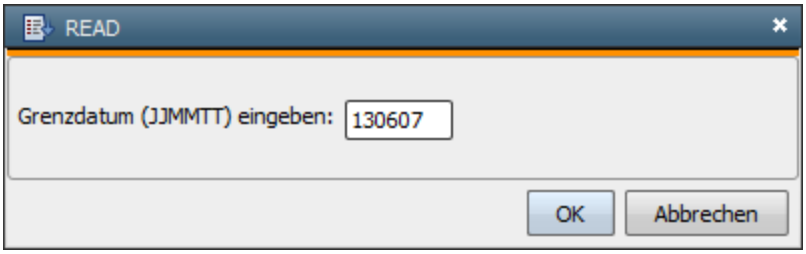
Abfrage einer Zahl. Der Parameter "N" erzeugt in diesem Fall ein Zahlenfeld mit Pfeilen. Es können nur Zahlen im Bereich von 0 bis 5 eingetragen werden.

```
:READ &ANZAHL#,"0-5","Anzahl eingeben",,"N"
```



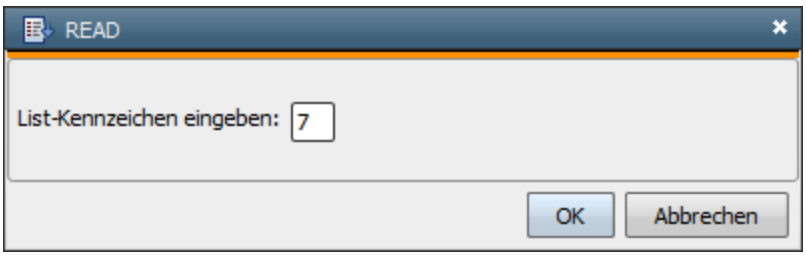
Abfrage eines Datums vom Benutzer. Der eingelesene Wert muss im Format "JJMMTT" ein gültiges Datum sein. Es handelt sich um eine Muss-Eingabe.

```
:READ &DATUM1#,"JJMMTT","Grenzdatum (JJMMTT) eingeben",,"M"
```



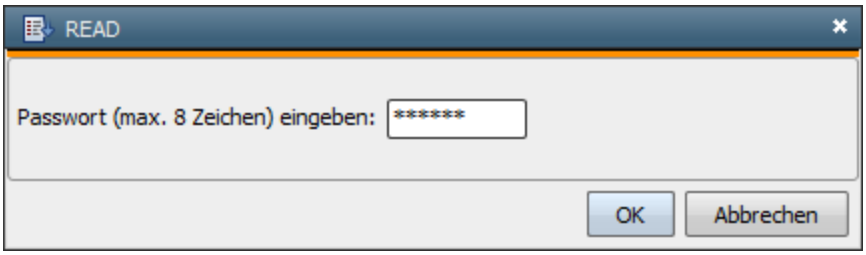
Abfrage eines Kennzeichens vom Benutzer. Das Eingabefeld enthält als Vorgabewert "A". Der eingelesene Wert wird überprüft und darf nur "A", "X", "5", "6", "7", "8" oder "9" sein.

```
:READ &LKZ#,"A,X,5-9","List-Kennzeichen eingeben","A"
```



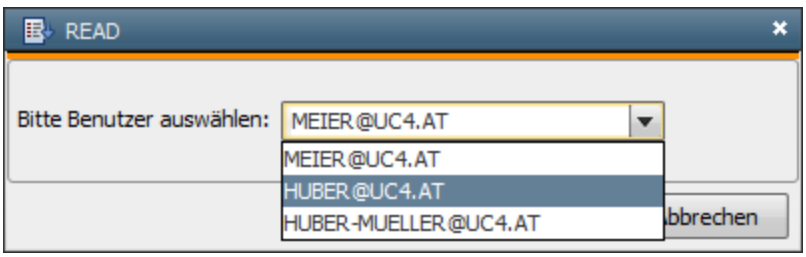
Abfrage eines Passwortes vom Benutzer mit maximal 8 Zeichen. Die *Eingabeformatierung* legt fest, dass die Eingabe geschützt erfolgt, dass keine Umsetzung auf Großbuchstaben erfolgt und dass eine Eingabe vorgeschrieben ist.

```
:READ &PASS#,"08","Passwort (max. 8 Zeichen) eingeben",,"DMK"
```



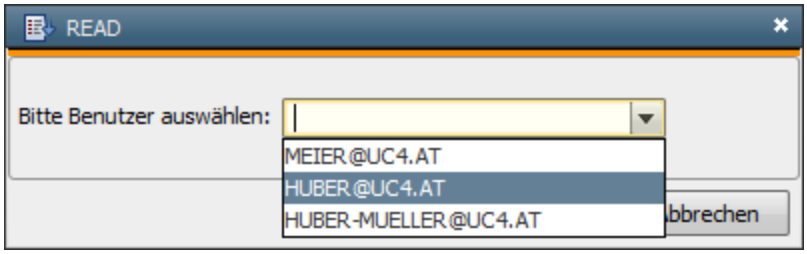
Im Beispiel wird die E-Mail-Adresse abgefragt. Der Benutzer kann eine von drei auswählen. Eine der E-Mail-Adressen ist unter einfache Hochkommata gesetzt, damit der darin enthaltene Bindestrich nicht als Wertebereich interpretiert wird.

```
:READ  &BENUTZER#,"meier@automic.at, huber@automic.at,'huber-
mueller@automic.at'", "Bitte Benutzer auswählen"
```



Im folgenden Beispiel kann der Benutzer nicht nur eine der drei E-Mail-Adressen auswählen, sondern auch selbst eine eingeben.

```
:READ  &BENUTZER#,"meier@automic.at, huber@automic.at,'huber-
mueller@automic.at'", "Bitte Benutzer auswählen",, "O"
```



Siehe auch:

:BEGINREAD... :ENDREAD	Beginn und Ende eines Dialogfeldes für Benutzerabfragen.
:PRINT	Dient zur Ausgabe von Text in einem Dialogfeld für Benutzerabfragen oder im Aktivierungsprotokoll eines Objektes.
:PUT_READ_ BUFFER	Stellt Namen und Inhalt einer Script-Variablen in den Zwischenspeicher.

Script-Sprachmittel - Objekte aktivieren
Beispielsammlung:
Datenbankwartung mit Optionen
Benachrichtigung mit variablem Meldungstext
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.3.5 ACTIVATE_UC_OBJECT

Script-Funktion: Aktiviert ein Objekt.

Syntax

***ACTIVATE_UC_OBJECT(Objektname** [, [**WAIT**] [, [logischesDatum] [, [Zeitzone] [, , [**PASS_VALUES**] [, [Queue] [, [Alias] [, [**ENABLE_PROMPTS**] [, Zeitlimit , [**TERMINATE**][, Ausführen]]]]]]]]])*
***ACTIVATE_UC_OBJECT(Objektname** [, , [logischesDatum] [, [Zeitzone] [, [Startzeitpunkt] [, [**PASS_VALUES**] [, [Queue] [, [Alias] [, [**ENABLE_PROMPTS**]]]]]]]])*

Syntaxteil	Beschreibung/Format
Objektname	Name des Objektes Format: AE-Name, Script-Literal oder Script-Variable
WAIT	Das Schlüsselwort WAIT bewirkt, dass die Script-Funktion auf die Beendigung des Objektes wartet.

logischesDatum	Logisches Datum, mit dem das Objekt aktiviert werden soll im Format "JJMMTT" oder "JJJJMMTT". Format: Script-Literal oder Script-Variable Es ist möglich das Datum in einem anderen Datumsformat anzugeben. Geben Sie dazu zuerst das gewünschte Datumsformat, ein Trennzeichen (: oder ;) und danach das Datum an. Die Angabe des Datumsformats ist optional.
Zeitzone	Name eines Zeitzonen-Objektes. Zeitzone, für die der Zeitstempel umgerechnet werden soll. Format: Script-Literal oder Script-Variable
Startzeitpunkt	Zeitstempel aus Datum und Uhrzeit ("JJJJ-MM-TT HH:MM:SS"). Format: Script-Literal oder Script-Variable
PASS_VALUES	Das Schlüsselwort PASS_VALUES bewirkt, dass die Objektvariablen und PromptSet-Variablen an die zu startende Aufgabe vererbt werden.
Queue	Angabe eines bestimmten Queue-Objektes, das für den Start der Aufgabe verwendet werden soll. Format: AE-Name, Script-Literal oder Script-Variable Erfolgt keine Queue-Angabe, wird das Objekt standardmäßig in der Mandanten-Queue (CLIENT_QUEUE) aktiviert.
Alias	Alias für das zu aktivierende Objekt. Dieser wird im Aktivitätenfenster und in der Statistik statt des Objektnamens angezeigt. Format: AE-Name, Script-Literal oder Script-Variable Der Alias unterliegt den selben Regeln wie Objektnamen in der Automation Engine: Maximale Länge: 200 Zeichen Erlaubte Zeichen: A-Z, 0-9, $, @, _, . und #
ENABLE_ PROMPTS	Aufruf der PromptSet-Eingabemasken des aktivierten Objektes im UserInterface Voraussetzungen: 1) Ein Benutzer ist am UserInterface angemeldet 2) Das Sprachmittel muss sich in einem Objekt befinden, das durch diesen Benutzer gestartet wird 3) Der Benutzer muss beim Sprachmittel-Aufruf immer noch mit der selben Session-ID am UserInterface angemeldet sein. Sind die oberen Voraussetzungen nicht erfüllt oder dem zu aktivierenden Objekt keine PromptSet-Objekte zugeordnet, so hat dies keine Auswirkung auf die weitere Ausführung des Sprachmittels. Die Eingabedialoge werden einfach nicht angezeigt.

Zeitlimit	Maximale Laufzeit des gestarteten Objektes Format: Script-Literal oder Script-Variable Die Angabe muss im Zeitformat HH:MM:SS erfolgen. MM und SS sind zweistellig anzugeben, HH kann auch einstellig sein. Beispiel: "1:05:07" Beachten Sie, dass dieser Parameter nur im Zusammenhang mit dem Schlüsselwort **WAIT** verwendet werden kann. Wird die maximale Laufzeit definiert, so ist auch zumindest eine Aktion festzulegen, die im Überschreitungs-Fall durchgeführt wird. Verwenden Sie dazu die Parameter **TERMINATE** und *Ausführen*.
TERMINATE	Aufgabe abbrechen, wenn die maximale Laufzeit (Zeitlimit) überschritten wird
Ausführen	Objekt, welches aktiviert werden soll, wenn die maximale Laufzeit überschritten wird. Format: AE-Name, Script-Literal oder Script-Variable

Rückgabewerte
Laufende Nummer (RunID) des aktivierten Objektes. "0" - Die Aktivierung war nicht erfolgreich. "20423" - Die Aufgabe versuchte sich selbst zu aktivieren. Das ist nicht erlaubt, weil eine Endlosschleife die Folge wäre.

Anmerkungen

Mit der Script-Funktion ACTIVATE_UC_OBJECT können Sie ein Objekt aktivieren, das der Objektklasse der aktivierbaren Objekte angehört.

Geben Sie keinen bestimmten Startzeitpunkt an, so erfolgt die Aktivierung des Objektes sofort. Bei der Festlegung eines bestimmten *Startzeitpunktes* wird die Aufgabe eingeplant und erhält den Status "Wartet auf Startzeitpunkt".

⚠ Denken Sie daran, dass Aktivierungs- und Startzeitpunkt nicht übereinstimmen müssen!

Das Objekt kann zusätzlich mit einem logischen Datum aktiviert werden. Fehlt dabei das Datumsformat, müssen Sie das Datum in den Formaten "JJMMTT" oder "JJJJMMTT" angeben. Als Trennzeichen zwischen Datumsformat und *Datum* sind Doppelpunkt oder Semikolon erlaubt.

Wird das Schlüsselwort WAIT verwendet, kann zusätzlich eine maximale Laufzeit für das gestartete Objekt definiert werden (*Zeitlimit*). Wird diese überschritten, so wird die Aufgabe abgebrochen und / oder ein zusätzliches beliebiges Objekt ausgeführt. Dabei ist zumindest eine dieser beiden Aktionen über die Parameter TERMINATE und *Ausführen* zu definieren.

Die Script-Funktion liefert nach einer erfolgreichen Aktivierung die RunID des Objektes. Bei Fehlern ist der Rückgabewert "0".

Fragen Sie immer den Rückgabewert ab. Mit der Script-Anweisung :ON_ERROR und den Script-Funktionen zur Fehlerbehandlung lässt sich die genaue Fehlerursache analysieren.

Situation	Rückgabewert	Beschreibung
Allgemein		

Objekt wurde erfolgreich aktiviert	RunID	Dies ist der Normalfall.
Objekt existiert nicht	0	Das Objekt ist nicht auffindbar und kann daher auch nicht aktiviert werden.
Kein Ausführrecht für das Objekt	0	Das Berechtigungssystem prüft ob der Benutzer, der für die Aktivierung verantwortlich ist, auch ein Ausführrecht für das Objekt besitzt.
Abbruch der Aktivierung des Objektes	0	Der Abbruch kann manuell oder durch andere Aufgaben erfolgen.
Der Startzeitpunkt liegt in der Vergangenheit	RunID	Ein Statistikeintrag "ENDED_TIMEOUT - Startzeit überschritten" wird erstellt.
Script-Generierung bei der Aktivierung		
Scriptfehler im Objekt	0	Scriptfehler brechen die Aktivierung eines Objektes ab.
:EXIT 0	RunID	Bei dieser Anweisung handelt es sich um eine Beendigung der Script-Prozessierung und nicht um einen Abbruch. Sie wird daher nicht als Fehler gewertet.
:EXIT <> 0	0	Diese Anweisung bricht die Aktivierung des Objektes ab.
:STOP NOMSG	RunID	Bei dieser Anweisung handelt es sich um eine Beendigung der Script-Prozessierung und nicht um einen Abbruch. Sie wird daher nicht als Fehler gewertet.
:STOP MSG	0	Diese Anweisung bricht die Aktivierung des Objektes ab.
Script-Generierung zur Laufzeit		
Scriptfehler im Objekt :EXIT 0 :EXIT <> 0 :STOP MSG :STOP NOMSG	RunID	Ist bei dem zu aktivierenden Objekt die Option "Generieren zur Laufzeit" gesetzt, so findet die Script-Ausführung nicht während der Aktivierung statt und ACTIVATE_UC_OBJECT kann eventuell auftretende Fehler nicht erkennen. In diesen Fällen wird immer die RunID als Rückgabewert geliefert.
Post-Script		
Scriptfehler im Objekt	RunID	Das Post-Script wird erst am Ende der Durchführung prozessiert und liegt daher nicht im Bereich der Aktivierung. Fehler werden daher nicht erkannt.

Ein Objekt kann sich über die Script-Funktion nicht selbst aktivieren. Damit werden Endlosschleifen vermieden, die das ganze Automation Engine System lahmlegen könnten.

⚠ Beachten Sie, dass der Parameter WAIT nicht zusammen mit der Angabe eines Startzeitpunktes verwendet werden kann.

Die Parameter der Script-Funktion sind in einer bestimmten Reihenfolge anzuordnen. Wenn Sie daher Parameter auslassen, sind trotzdem deren Kommata zu setzen (siehe nachfolgende Beispiele).

Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

Beispiele

Im ersten Beispiel wird ein Job "Status" aktiviert, wobei eine Kontrolle des Rückgabewertes erfolgt.

```
:SET  &ACTOBJ# = ACTIVATE_UC_OBJECT(STATUS)

:IF  &ACTOBJ# = "0"
:    SET  &ERRNR# = SYS_LAST_ERR_NR()
:    SET  &ERRINS#  = SYS_LAST_ERR_INS()
:    SET  &MESSAGE# = GET_MSG_TXT(&ERRNR#,&ERRINS#)
:    SET  &RET# = SEND_MAIL("hans.meier@automic.com",,&MESSAGE#, "Bitte
prüfen. Danke!")
:ENDIF
```

Das zweite Beispiel aktiviert den Job mit einem logischen Datum.

```
:SET  &ACTOBJ# = ACTIVATE_UC_OBJECT(STATUS,,"DD.MM.YY:01.12.00")
```

Der Workflow MAWI.WOCHE soll Mitte Juli am Abend laufen.

```
:SET  &ACTOBJ# = ACTIVATE_UC_OBJECT(MAWI.WOCHE,,, "MEZ", "2005-07-15
18:00:00")
```

An den Job MAWI.ABSCHLUSS sollen auch die Objektvariablen vererbt werden.

```
:SET  &ACTOBJ# = ACTIVATE_UC_OBJECT("MAWI.ABSCHLUSS",,,,, PASS_VALUES)
```

Siehe auch:

Script-Sprachmittel	Beschreibung
RESTART_UC_OBJECT	Wiederholt die Durchführung einer Aufgabe.
CANCEL_UC_OBJECT	Bricht ein aktiviertes Objekt ab.
GET_UC_OBJECT_NR	Liefert die laufende Nummer (RunID) eines aktivierten Objektes.
CREATE_OBJECT	Erstellt ein Objekt (nur Kalender, Login und Variable).
:ON_ERROR	Legt die Reaktion auf bestimmte Fehler bzw. Meldungen von Script-Sprachmitteln fest.

Script-Sprachmittel - Objekte aktivieren

Beispielsammlung:

Datenbankwartung mit Optionen
Benachrichtigung mit variablem Meldungstext
Ermittlung der Fehlermeldung und -nummer
Reaktion auf externe Ereignisse

Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.3.6 AUTOFORECAST

Script-Funktion: Berechnet Prognosedaten für zukünftige Aktivitäten.

Syntax

AUTOFORECAST*()*

Rückgabewert
"0" - Die Prognosedaten wurden erfolgreich angelegt.

Anmerkungen

Die Autoprognose zeigt jene Aufgaben an, die in einem vorgegebenen Zeitraum laufen werden, und bietet somit eine umfassende Vorschau auf zukünftige Aktivitäten. Die Berechnung der Prognosedaten kann manuell im UserInterface erfolgen oder mit der Script-Funktion AUTOFORECAST. Dabei werden Prognosen für die aktiven Schedules und Ereignisse angelegt.

Die Script-Funktion berechnet nur Prognosedaten für den Mananten in dem sie aufgerufen wird.

Mit der Script-Anweisung :ON_ERROR legen Sie die Reaktion auf einen Fehler fest. Sie können ihn mit den Script-Funktionen zur Fehlerbehandlung analysieren. Das Script wird weiter prozessiert. Es besteht aber auch die Möglichkeit, die Prozessierung des Scripts abzubrechen.

Nähere Erläuterung zur Autoprognose finden Sie im gleichnamigen Dokument im Kapitel UserInterface.

Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

Beispiel

Das Beispiel zeigt die Anwendung der Script-Funktion.

```
:SET &RET# = AUTOFORECAST()
```

Siehe auch:

Script-Sprachmittel	Beschreibung
FORECAST_OBJECT	Erzeugt eine Prognose für das angegebene Objekt.
FORECAST_TASK	Erzeugt eine Prognose für die angegebene Aufgabe.

Script-Sprachmittel - Objekte aktivieren
Autoprognose

Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.3.7 CANCEL_UC_OBJECT

Script-Funktion: Bricht ein aktiviertes Objekt ab.

Syntax

CANCEL_UC_OBJECT(***RunID*** *[, Zusatz]*)

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) des aktivierten Objektes Format: Script-Literal oder Script-Variable
Zusatz	Dieser zusätzliche Parameter gilt für bestimmte Objekttypen. Format: AE-Name oder Script-Literal • Für Ereignisse: Bei Ereignis-Objekten können Sie einen Status angeben mit dem das Ereignis-Objekt beendet wird. Erlaubte Werte: "ENDED_OK" "ENDED_CANCEL" "ENDED_TIMEOUT" • Für Workflows, Schedules und Gruppen: Bei diesen Objekttypen bewirkt die Verwendung des Parameters "ALL", dass auch alle laufenden untergeordneten Aufgaben abgebrochen werden. Erlaubter Wert: "ALL"

Rückgabewerte
"0" - Die Aufgabe wurde erfolgreich abgebrochen. "11049" - Die Aufgabe mit der RunID wurde für den Abbruch nicht gefunden. "11050" - Die Aufgabe befindet sich in einem nicht abbrechbaren Zustand. "20347" - Ein Script, das über ein CallAPI gestartet worden ist, kann nicht mit CANCEL_UC_OBJECT abgebrochen werden.

Anmerkungen

Mit der Script-Funktion CANCEL_UC_OBJECT können Sie alle aktivierbaren Objekte abbrechen. Der Zugriff auf das aktivierte Objekt erfolgt über die Kurzbezeichnung des Objekttyps und die laufende Nummer (RunID), mit der dieses aktiviert wurde.

Mit der Script-Anweisung :ON_ERROR legen Sie die Reaktion auf Fehler fest. Wie bisher können Sie sie mit den Script-Funktionen zur Fehlerbehandlung analysieren. Das Script wird weiter prozessiert. Es besteht aber auch die Möglichkeit, die Prozessierung des Scripts abzubrechen.

Bei einem Ereignis beachten Sie bitte folgende Besonderheit: Bei jedem Eintreten des Ereignisses wird eine neue Instanz des aktivierten Ereignisses erzeugt. Diese Instanz erhält auch eine eigene laufende Nummer (RunID). Wollen Sie ein Ereignis aufgrund eingetretener Bedingungen abbrechen, müssen Sie die *laufende Nummer (RunID)* mit der Funktion SYS_ACT_PARENT_NR ermitteln.

Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

In früheren Versionen konnten Sie die Script-Funktion CANCEL_UC_OBJECT dazu verwenden, um beendete Aufgaben zu deaktivieren. Dies wurde geändert, es steht Ihnen jetzt hierfür die Script-Funktion DEACTIVATE_UC_OBJECT zur Verfügung.

Beispiele

Im Beispiel wird das Benachrichtigungs-Objekt "ALARMIERUNG3" aktiviert. Die :READ-Anweisung dient nur dazu solange zu warten, bis die Benachrichtigung angezeigt wird. Nach dem Einlesen der laufenden Nummer (RunID) und der Bestätigung durch den Benutzer wird das Benachrichtigungs-Objekt abgebrochen.

```
:SET &JOBNR# = ACTIVATE_UC_OBJECT(ALARMIERUNG3)
:READ &JOBNR#,,,&JOBNR#
:SET &STATUS# = CANCEL_UC_OBJECT(&JOBNR#)
```

Das Beispiel zeigt einen Auszug aus dem Script eines Ereignisses. Bei der Prozessierung des Scripts zum Ereigniseintritt wird die laufende Nummer (RunID) ermittelt und das Ereignis abgebrochen.

```
:SET &JOBNR# = SYS_ACT_PARENT_NR()
:SET &STATUS# = CANCEL_UC_OBJECT( &JOBNR#)
```

Die folgenden Zeilen brechen ein laufendes Ereignis-Objekt mit dem Status "ENDED_CANCEL" ab.

```
:SET &RUNNR# = GET_UC_OBJECT_NR(EVNT.NACHT)
:SET &STATUS# = CANCEL_UC_OBJECT(&RUNNR#, "ENDED_CANCEL")
```

Siehe auch:

Script-Sprachmittel	Beschreibung
ACTIVATE_UC_OBJECT	Aktiviert ein Objekt.
RESTART_UC_OBJECT	Wiederholt die Durchführung einer Aufgabe.
GET_UC_OBJECT_NR	Liefert die laufende Nummer (RunID) eines aktivierten Objektes.
:ON_ERROR	Legt die Reaktion auf bestimmte Fehler bzw. Meldungen von Script-Sprachmitteln fest.

Script-Sprachmittel - Objekte aktivieren

Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.3.8 DEACTIVATE_UC_OBJECT

Script-Funktion: Deaktiviert eine abgeschlossene Aufgabe.

Syntax

DEACTIVATE_UC_OBJECT*(RunID* [, FORCED]*)*

Syntaxteil	Beschreibung/Format
RunID	RunID jener Aufgabe, die deaktiviert werden soll. Format: Script-Variable, Script-Literal, Zahl ohne Hochkommata
FORCED	Deaktivierung unabhängig vom Status der übergeordneten Aufgaben durchführen Format: AE-Name, Script-Literal oder Variable

Rückgabewert
12204 - Aufgabe mit RunID kann nicht deaktiviert werden. Objekte eines Workflows können nicht deaktiviert werden. 12205 - Deaktivieren von Aufgabe(RunID) nicht möglich, da noch Subaufgaben aktiv sind. 12206 - Die Zustandsdefinition ist nicht numerisch 12207 - Die Zustandsdefinition nicht aufsteigend (von-bis) 12208 - Die Zustandsdefinition ist syntaktisch falsch 12209 - Zustandsdefinition nicht gefunden: Objekt wurde nicht automatisch deaktiviert 12210 - Zustandsdefinition nicht numerisch, Objekt wurde nicht automatisch deaktiviert

Anmerkungen

Deaktivieren ist nur für bereits beendete Aufgaben (erfolgreich oder abgebrochen) möglich und entfernt diese aus den Aktivitäten (Aktivitätenfenster). Dazu ist die RunID der Aufgabe anzugeben.

Für die Deaktivierung gelten folgende Voraussetzungen:

- Wird ein Workflow angegeben, darf dieser keine untergeordneten Aufgaben enthalten, die noch aktiv sind. Dies gilt auch für die Aufgaben aller Sub-Workflows.
- Aufgaben, die innerhalb eines Workflows laufen, können nicht deaktiviert werden. Dies ist nur durch Deaktivierung des obersten Workflows möglich (für diesen gilt allerdings wieder obige Voraussetzung).

In ausführbaren Objekten können auch Einstellungen zur automatischen Deaktivierung gesetzt werden. Diese werden ebenfalls geprüft.

Wird zusätzlich das Schlüsselwort FORCED angegeben, so wird nicht geprüft, ob die Aufgabe innerhalb eines Workflows oder innerhalb eines Sub-Workflows läuft. Weiters werden die Optionen zur automatischen Deaktivierung aller untergeordneten Aufgaben ignoriert und diese trotzdem deaktiviert. FORCED kann nur in Zusammenhang mit Workflows verwendet werden.

⚠ Beachten Sie: "Deaktivieren (erzwungen)" prüft nicht, ob Sub-Workflows noch aktive Aufgaben enthalten.

Beispiel

In folgendem Beispiel wird ein Objekt aktiviert und auf dessen Ende gewartet. Im Fehlerfall wird anschließend eine Meldung an den betreffenden Benutzer gesendet. Endet das Objekt ohne Fehler, so wird es deaktiviert.

```
:SET &ACTOBJ# = ACTIVATE_UC_OBJECT(&OBJ#,WAIT)

:IF &ACTOBJ# = "0"
:   SET &ERRNR# = SYS_LAST_ERR_NR()
:   SET &ERRINS#  = SYS_LAST_ERR_INS()
:   SET &MESSAGE# = GET_MSG_TXT(&ERRNR#,&ERRINS#)
:   SET &RET# = SEND_MAIL("hans.meier@uc4.com",,&MESSAGE#, "Bitte
prüfen. Danke!")
:ELSE
: SET &DEACTJOB# = DEACTIVATE_UC_OBJECT(&ACTJOB#)
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
ACTIVATE_UC_OBJECT	Aktiviert ein Objekt.
RESTART_UC_OBJECT	Wiederholt die Durchführung einer Aufgabe.

Script-Sprachmittel - Objekte aktivieren
Beispielsammlung:
Datenbankwartung mit Optionen
Benachrichtigung mit variablem Meldungstext
Ermittlung der Fehlermeldung und -nummer
Reaktion auf externe Ereignisse
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.3.9 FORECAST_OBJECT

Script-Funktion: Erzeugt eine Prognose für das angegebene Objekt.

Syntax

FORECAST_OBJECT*(Objektname, Titel, Datum, Startzeit *[,Vormerkungen] [,Gruppen_ERT]**)***

Syntaxteil	Beschreibung/Format
Objektname	Name des Objektes. Format: Script-Literal oder Script-Variable

Titel	Name für die Prognose. Format: Script-Literal oder Script-Variable
Datum	Logisches Startdatum im Format "JJMMTT" oder "JJJJMMTT" Format: Script-Literal oder Script-Variable Es ist möglich, das Datum in einem anderen Datumsformat anzugeben. Geben Sie dazu zuerst das gewünschte Datumsformat, ein Trennzeichen (: oder ;) und danach das Datum an. Die Angabe des Datumsformats ist optional.
Startzeit	Startzeitpunkt der Aufgabe im Format "HHMMSS". Format: Script-Literal oder Script-Variable Es ist möglich, die Zeit in einem anderen Zeitformat anzugeben. Geben Sie dazu zuerst das gewünschte Zeitformat, ein Trennzeichen (;) und danach die Zeit an. Die Angabe des Zeitformats ist optional.
Vormerkungen	Einstellung, ob die Vormerkungen (bei Gruppen) berücksichtigt werden sollen. Format: Script-Literal oder Script-Variable Erlaubte Werte: "Y", "N" (Standardwert)
Gruppen_ERT	Erwartete Laufzeit (ERT) bei Gruppen. Wird verwendet, wenn die *Vormerkungen* auf 'N' gesetzt werden. Format: Script-Variable oder Zahl Erlaubte Werte: "0" - ERT der Gruppe wird verwendet. (Standardwert) "1" bis "7199" - Fixwert in Sekunden

Rückgabewert
"0" - Die Prognose konnte erfolgreich angelegt werden.

Anmerkungen

Für jedes aktivierbare Objekt ist es möglich eine Prognose über die Laufzeit zu erstellen. Dies kann einerseits im UserInterface selbst gemacht werden, indem über das Kontextmenü des Explorers der gleichnamige Befehl aufgerufen wird. Unter der Angabe bestimmter Kriterien wird für das Objekt eine Prognose angelegt. Die zweite Möglichkeit ist die Benützung dieser Script-Funktion.

Ähnlich der manuellen Eingabe sind ein Name für die Prognose (in Großbuchstaben, ohne Leerzeichen!) sowie Startdatum- und -zeit anzugeben. Nutzen Sie die Möglichkeit für letztere ein Format Ihrer Wahl zu definieren. Für Aufgaben, die unter anderem einer Gruppe zugehörig sind, können die Parameter *Vormerkungen* und *Gruppen_ERT* nach Bedarf eingestellt werden. Beachten Sie, dass der Titel für die Prognose noch nicht existieren darf!

Verwenden Sie Für die Erstellung einer Prognose von laufenden Aufgaben das Script-Sprachmittel FORECAST_TASK.

Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

Beispiel

Die folgende Zeile erzeugt einen Forecast für einen FileTransfer unter der zusätzlichen Verwendung von Datums- und Zeitformaten.

```
:SET &RET# = FORECAST_OBJECT("C70.WINDOWS","FORECAST_FT_
C70.WINDOWS","MM/DD/YY:11/03/03","HH:MM;12:30")
```

Im zweiten Beispiel wird die Prognose für einen Job erstellt, der innerhalb einer Gruppe läuft. Dabei soll nicht die ERT von dieser verwendet werden, sondern ein Fixwert von 60 Sekunden.

```
:SET &RET# = FORECAST_OBJECT("T91.SALDO.03","PROGNOSE_
T91.SALDO.03","MM/DD/YY:11.07/03","200000","Y",60)
```

Siehe auch:

Script-Sprachmittel	Beschreibung
AUTOFORECAST	Berechnet Prognosedaten für zukünftige Aktivitäten.
FORECAST_TASK	Erzeugt eine Prognose für die angegebene Aufgabe.

Script-Sprachmittel - Objekte aktivieren
Prognose
Laufzeitermittlung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.3.10 FORECAST_TASK

Script-Funktion: Erzeugt eine Prognose für die angegebene aktive Aufgabe.

Syntax

FORECAST_TASK*(RunID, Titel *[,Vormerkungen] [,Gruppen_ERT] [, Tage]* **)**

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) des aktivierten Objektes Format: Script-Literal oder Script-Variable
Titel	Name für die Prognose. Format: Script-Literal oder Script-Variable
Vormerkungen	Einstellung, ob die Vormerkungen (bei Gruppen) berücksichtigt werden sollen. Format: Script-Literal oder Script-Variable Erlaubte Werte: "Y", "N" (Standardwert)

Gruppen_ERT	Erwartete Laufzeit (ERT) bei Gruppen. Wird verwendet, wenn die *Vormerkungen* auf 'N' gesetzt werden. Format: Zahl ohne Hochkommata, Script-Literal oder Script-Variable Erlaubte Werte: "0" - ERT der Gruppe wird verwendet. (Standardwert) "1" bis "7199" - Fixwert in Sekunden
Tage	Anzahl der Tage, die maximal in die Zukunft gerechnet werden soll Format: Zahl ohne Hochkommata, Script-Literal oder Script-Variable Standardwert: "1"

Rückgabewerte
"0" - Die Prognose konnte erfolgreich angelegt werden. "20463" - Laufzeitfehler in Objekt '&01' Zeile '&02': Ungültige Syntax in Bedingung.

Anmerkungen

Für jede laufende Aufgabe ist es möglich eine Prognose über die Laufzeit zu erstellen. Dies kann einerseits im UserInterface selbst gemacht werden, indem über das Kontextmenü des Aktivitätenfensters der gleichnamige Befehl aufgerufen wird. Unter der Angabe bestimmter Kriterien wird für die Aufgabe eine Prognose angelegt. Die zweite Möglichkeit ist die Benützung dieser Script-Funktion.

Genauso wie bei der manuellen Eingabe ist ein Name für die Prognose (in Großbuchstaben, ohne Leerzeichen!) anzugeben. Für Aufgaben, die unter anderem einer Gruppe zugehörig sind, können die Parameter *Vormerkungen* und *Gruppen_ERT* nach Bedarf eingestellt werden. Beachten Sie, dass der Titel für die Prognose noch nicht existieren darf!

Verwenden Sie Für die Erstellung einer Prognose von Objekten das Script-Sprachmittel FORECAST_OBJECT.

Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

FORECAST_TASK funktioniert nur mit aktiven Aufgaben.

Beispiel

Die folgende Zeile erzeugt einen Forecast für einen FileTransfer.

```
:SET &RUNNR# = GET_UC_OBJECT_NR ("C70.WINDOWS")
:SET &RET# = FORECAST_TASK(&RUNNR#,"PROGNOSE_FT_C70.WINDOWS")
```

Im zweiten Beispiel wird die Prognose für einen Job erstellt, der innerhalb einer Gruppe läuft. Dabei soll nicht die ERT von dieser verwendet werden, sondern ein Fixwert von 60 Sekunden.

```
:SET &RUNNR# = GET_UC_OBJECT_NR ("T91.SALDO.03")
:SET &RET# = FORECAST_TASK(&RUNNR#,"PROGNOSE_T91.SALDO.03","N",60)
```

Siehe auch:

Script-Sprachmittel	Beschreibung

AUTOFORECAST	Berechnet Prognosedaten für zukünftige Aktivitäten.
FORECAST_OBJECT	Erzeugt eine Prognose für das angegebene Objekt.

Script-Sprachmittel - Objekte aktivieren
Prognose
Laufzeitermittlung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.3.11 RERUN_UC_OBJECT

Script-Funktion: Fortsetzen eines bestimmten Workflows.

Syntax

RERUN_UC_OBJECT*(RunID)*

Syntaxteil	Beschreibung/Format
RunID	RunID des Workflows, dessen Aufgaben fortgesetzt werden sollen. Format: Script-Variable, Zahl ohne Hochkommata oder Script-Literal

Rückgabewert
12113 - Bei der angegebenen Aufgabe handelt es sich nicht um einen Workflow. Ein Rerun ist lediglich auf einen Workflow möglich. 20282 - Objekt mit RunID nicht gefunden.

Anmerkungen

Bei "Fortsetzen" handelt es sich um einen speziellen Vorgang, der für aktive Workflows durchgeführt werden kann und im Zusammenhang mit der Rollback-Funktionalität steht. Durch Fortsetzen eines Workflows werden alle seine Child-Aufgaben mit dem Status ENDED_ ROLLBACKED, ENDED_ROLLBACK_EMPTY und / oder "Wartet auf Rollback" neu gestartet.

Beispiel

Folgendes Beispiel ermittelt den Status einen Jobs. Ist dieser 1904 (ENDED_ROLLBACKED), so wird der Workflow des Jobs fortgesetzt.

```
:SET &RET# = GET_UC_OBJECT_STATUS(, &RUNID#, "STATUS")

:IF &RET# = 1904
:SET &PARENT# = GET_PARENT_NR(&RUNID#)
:SET &RB# = RERUN_UC_OBJECT(&PARENT#)
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
ROLLBACK_UC_OBJECT	Rollback einer bestimmten Aufgabe durchführen.

Script-Sprachmittel - Datensequenzen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.3.12 RESTART_UC_OBJECT

Script-Funktion: Wiederholt die Durchführung einer Aufgabe.

Syntax

RESTART_UC_OBJECT (*Objektname, Referenz-RunID,* *[Wiederanlauf-Punkt]*, *[Flags]* *[, Queue]* *)*

Syntaxteil	Beschreibung/Format
Objektname	Name des Objektes Format: Script-Literal, Script-Variable oder Script-Funktion
Referenz-RunID	RunID, auf die sich der Wiederanlauf bezieht, oder das Schlüsselwort **LAST** für die letzte Durchführung der Aufgabe. Format: Script-Literal, Zahl, Script-Variable oder Script-Funktion Die Wiederholung einer Aufgabe, die selbst ein Wiederanlauf war, ist nicht erlaubt. Die Nachvollziehbarkeit der Aktivitäten wäre dadurch nicht gewährleistet. Deshalb muss sich die RunID auf eine Original-Durchführung beziehen. Bei LAST wird die letzte Original-Durchführung verwendet.
Wiederanlauf-Punkt	Stelle im Script, ab der die Verarbeitung wiederholt werden soll. Format: AE-name, Script-Literal, Zahl, Script-Variable Die Angabe dieses Parameter ist natürlich nur möglich, wenn Sie in der Aufgabe, die wiederholt werden soll, im Script Wiederanlaufpunkte mit :RESTART gesetzt haben. Verwenden Sie diesen Parameter nicht, so wird das gesamte Script durchgeführt.

Flags	Anweisungen für die Durchführung der Aufgabe. Format: Script-Literal oder Script-Variable Erlaubte Werte: "GEN_JCL" = Die generierte JCL im Aktivierungsreport ausgeben. "ORIGINAL_SCRIPT" = Das Original-Script im Aktivierungsreport ausgeben. "VAR_MOD" = Veränderung von Variablen im Aktivierungsreport ausgeben. "ATT_MOD" = Veränderung von Attributen im Aktivierungsreport ausgeben. "ATT_DIALOG" = Attributdialog aktivieren. "MAN_RELEASE" = Auf manuelle Freigabe warten. "KEEP_STARTTYPE" = Starttyp beibehalten. "ONLY_ABENDED" = Nur die untergeordneten Aufgaben, die abgebrochen sind, wiederholen. Es können mehrere *Flags*, durch Komma getrennt, angegeben werden. Wenn Sie diese namentlich angeben und keine Script-Variable verwenden, muss der gesamte Ausdruck in Hochkommata stehen. Die erlaubte Werte entsprechen den Optionen, die auch beim manuellen Wiederanlauf im Dialog "Ausführen..." vorhanden sind.
Queue	Angabe eines bestimmten Queue-Objektes, das für den Wiederanlauf der Aufgabe verwendet werden soll. Erfolgt keine Queue-Angabe, wird die Aufgabe automatisch in der Mandanten-Queue (CLIENT_QUEUE) neu gestartet.

Rückgabewert
"0" - Der Wiederanlauf der Aufgabe war erfolgreich. "7014" - Die Referenz-RunID existiert nicht. "7015" - Der Wiederanlauf einer Wiederanlaufdurchführung ist nicht erlaubt. "20346" - Der Wiederanlauf-Punkt existiert nicht. "20380" - Script-Prozessierung der Wiederanlauf-Durchführung wurde durch das Sprachmittel :EXIT beendet. "20385" - Beim Objekt handelt es sich um einen Schedule. "20628" - Das Objekt existiert nicht.

Anmerkungen

Mit der Script-Funktion kann ein Wiederanlauf einer Aufgabe durchgeführt werden.

Beachten Sie, dass sich eine Aufgabe auch selbst wiederholen kann, wenn die Script-Funktion im Post-Script aufgerufen wird. Dabei kann es jedoch zu einer Endlosschleife kommen!

Wenn Sie eine der vier Testoptionen als Flags wählen ("GEN_JCL", "ORIGINAL_SCRIPT", "VAR_MOD" oder "ATT_MOD"), wird der Job mit :STOP beendet. Grund dafür ist der Standardwert "N" (=Nein), der als Antwort auf die Frage "Soll die Aufgabe gestartet werden?" angegeben wird.

Um einen Restart erfolgreich mit allen oder einer der vier Testoptionen als Flags auszuführen, nutzen Sie anstelle des Scripts das UserInterface, beschrieben im Kapitel "Ausführen mit Optionen".

Der Script-Funktion wird mit dem Parameter *Referenz-RunID* die RunID der Aufgabe übergeben, deren Durchführung wiederholt werden soll. Es kann auch das Schlüsselwort LAST verwendet werden, um die letzte Durchführung einer Aufgabe erneut zu starten.

Optional kann ein Wiederanlauf-Punkt benannt werden, der bei der erneuten Durchführung der Aufgabe berücksichtigt werden soll. Die Aufgabe wird gestartet, aber die Verarbeitung des Scriptes beginnt erst ab dieser Stelle. Die Flags sind ebenfalls optional und beeinflussen die Durchführung der Aufgabe.

Mit der Script-Anweisung :ON_ERROR legen Sie die Reaktion auf Fehler fest. Wie bisher können Sie sie mit den Script-Funktionen zur Fehlerbehandlung analysieren. Das Script wird weiter prozessiert. Es besteht aber auch die Möglichkeit, die Prozessierung des Scripts abzubrechen.

Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

Beispiel

Im Beispiel soll die letzte Durchführung eines Jobs wiederholt werden. Die generierte JCL und die Veränderung von Variablen werden im Aktivierungsprotokoll ausgegeben. Der Job wartet im Aktivitätenfenster auf eine manuelle Freigabe.

```
:SET &RET# = RESTART_UC_OBJECT ("JOBS.SYSTEM.CHECK",LAST,,"GEN_
JCL,VAR_MOD,MAN_RELEASE")
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:GENERATE	Steuert die Behandlung von Script-Zeilen bei der Prozessierung eines Scripts.
:ON_ERROR	Legt die Reaktion auf bestimmte Fehler bzw. Meldungen von Script-Sprachmitteln fest.
:RESTART	Setzt Wiederanlauf-Punkte in einem aktivierbaren Objekt.
SYS_ACT_RESTART	Ermittelt, ob eine Aufgabe im Wiederanlauf-Modus aktiviert wurde.
SYS_ACT_RESTART_ME_NR	Liefert die laufende Nummer (RunID) des im Wiederanlauf-Modus aktivierten Objektes.
SYS_LAST_RESTART_POINT	Liefert die Bezeichnung des vorangegangenen Wiederanlauf-Punktes im Script.
SYS_LAST_RESTART_TEXT	Liefert den Text des vorangegangenen Wiederanlauf-Punktes im Script.
SYS_RESTART_POINT	Liefert den Wiederanlauf-Punkt, mit dem das Objekt ausgeführt wird.

Script-Sprachmittel - Objekte aktivieren
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.3.13 ROLLBACK_UC_OBJECT

Script-Funktion: Rollback einer bestimmten Aufgabe durchführen.

Syntax

ROLLBACK_UC_OBJECT*(RunID)*
ROLLBACK_UC_OBJECT*(*[Workflow-RunID]*, RunID2)*

Syntaxteil	Beschreibung/Format
RunID	RunID jener Aufgabe, für die das Rollback gestartet werden soll. Format: Script-Variable, Zahl ohne Hochkommata oder Script-Literal
RunID2	RunID der Aufgabe, bis zu der das Rollback innerhalb des Workflows durchgeführt werden soll. Format: Script-Variable, Zahl ohne Hochkommata oder Script-Literal Wird dieser Parameter angegeben, so ist der erste Parameter optional. Um den Workflow bis zum START-Objekt zurückzurollen, ist für *RunID2* das Schlüsselwort START (ohne Hochkommata) zu verwenden. In diesem Fall ist die *Workflow-RunID* unbedingt erforderlich!
Workflow-RunID	RunID des Workflows, in welchem das Rollback bis zur Aufgabe *RunID2* durchgeführt werden soll. Format: Script-Variable, Zahl ohne Hochkommata oder Script-Literal

Rückgabewert
11121 - Aufgabe mit der angegebenen RunID konnte nicht gefunden werden. 12108 - Rollback für Aufgabe nicht möglich. Rollback ist im Objekt nicht aktiviert. 12107 - Rollback für Aufgabe auf Grund des Status nicht möglich. 20282 - Laufzeitfehler: Objekt nicht gefunden.

Anmerkungen

Das Sprachmittel führt die Rollback-Aktionen der angegebenen Aufgabe durch. Welche Aktionen dabei durchgeführt werden, hängt von der Rollback-Definition im Objekt ab (Registerkarte **Rollback**).

⚠ Das Rollback kann nur für bereits abgeschlossene Workflow-Aufgaben gestartet werden, für welche diese Funktionalität aktiviert wurde.

Handelt es sich bei der angegebenen Aufgabe um einen Workflow, so wird das Rollback auch für alle untergeordneten Aufgaben durchgeführt.

Sie haben außerdem die Möglichkeit, das Rollback nur bis zu einer bestimmten Aufgabe innerhalb eines Workflows durchzuführen. Verwenden Sie dazu die RunID dieser Workflow-Aufgabe für den zweiten Parameter *RunID2*. Beim ersten Parameter kann die RunID des Workflows angegeben werden, in welchem die Aufgabe läuft. Diese Angabe ist aber optional.
Um den Workflow bis zum START-Objekt zurückzurollen, ist für *RunID2* das Schlüsselwort START zu verwenden. In diesem Fall ist die *Workflow-RunID* unbedingt anzugeben!

Genaue Informationen zur Durchführung von Backup und Rollback finden Sie im entsprechenden Dokument dazu.

Beispiel

Folgendes Beispiel aktiviert einen Job und überprüft dessen Rückgabewert. Endete die Aufgabe nicht normal, so wird deren Rollback gestartet.

```
:SET &ACT# = ACTIVATE_UC_OBJECT(JOBS.TEST,WAIT)
:SET &RET# = GET_UC_OBJECT_STATUS(, &ACT#, "RETCODE")

:IF &RET# > 0
: SET &RB# = ROLLBACK_UC_OBJECT(&ACT#)
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
RERUN_UC_OBJECT	Fortsetzen eines bestimmten Workflows.

Script-Sprachmittel - Datensequenzen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.3.14 SYS_STATE_ACTIVE

Script-Funktion: Prüft, ob ein Objekt bereits aktiv ist.

Syntax

SYS_STATE_ACTIVE(*[[Objekttyp,] Objektname]***)**

Syntaxteil	Beschreibung/Format
Objekttyp	Kurzbezeichnung eines aktivierbaren Objektes Format: AE-name, Script-Literal oder Script-Variable
Objektname	Name eines Objektes Format: AE-Name, Script-Literal oder Script-Variable

Rückgabewerte
"Y" - Das Objekt ist in einem Status mit System-Returncode < 1699 oder gleich 1701. "N" - Das Objekt ist in einem Status mit System-Returncode > 1699 und ungleich 1701.

Anmerkungen

Die Script-Funktion prüft, ob sich das angegebene Objekt, das der Objektklasse der aktivierbaren Objekte angehört, in einem Status mit System-Returncode kleiner 1699 befindet oder exakt dem Returncode 1701 entspricht. Werden beide Parameter nicht angegeben, kontrolliert die Funktion das Objekt in dem sie aufgerufen wird.

Die Prüfung erfolgt genau zu dem Zeitpunkt, an dem die Script-Zeile, welche die Funktion enthält, abgearbeitet wird. Ein negatives Ergebnis sagt nichts darüber aus, dass das zu prüfende Objekt nicht später aktiviert werden kann. Mit Hilfe der Funktionen kann daher keine Synchronisation von Abläufen erfolgen. Sie liefert nur eine zeitlich begrenzte Information über den Status von Objekten.

Wird der Name des eigenen Objektes angegeben (= Objekt, über welches das Sprachmittel ausgeführt wird), so wird dies auch berücksichtigt und immer der Rückgabewert "Y" zurückgeliefert.

Beispiele

Das Beispiel überprüft, ob das selbe Objekt bereits aktiv ist.

```
:SET  &AKTIV# = SYS_STATE_ACTIVE()
```

In diesem Beispiel wird die Funktion verwendet, um eine Bedingung zu formulieren.

```
:IF  SYS_STATE_ACTIVE(JOBS, "MAWI.ABSCHLUSS") = "Y"
!...
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_STATE_JOB_ACTIVE	Prüft, ob ein Job bereits aktiv ist.
SYS_STATE_JOBS_IN_GROUP	Ermittelt die Anzahl von Jobs, die in Gruppen vorgemerkt sind.
SYS_STATE_JP_ACTIVE	Prüft, ob ein Workflow bereits aktiv ist.

Script-Sprachmittel - Objekte aktivieren
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.3.15 SYS_ACTIVE_COUNT

Script-Funktion: Ermittelt die Anzahl der in den Aktivitäten stehenden Objekte.

Syntax

SYS_ACTIVE_COUNT*(Status, Objekttyp [,Objekt] [,Gruppe] [,Host])*

Syntaxteil	Beschreibung/Format

Status	Status der in den Aktivitäten befindlichen Objekte. Format: AE-name, Script-Literal oder Script-Variable Erlaubte Werte: "*", "ANY_ABEND", "BLOCKED", "ANY_ALIVE", "PREPARED" und "RUNNING" "*" = Aufgaben mit jedem Status "ANY_ABEND" = Abgebrochenen Aufgaben "ANY_ALIVE" = Aufgaben einer Gruppe, die nicht beendet oder abgebrochen sind "BLOCKED" = Blockierte Aufgaben "PREPARED" = Für eine Gruppe vorgemerkte Aufgaben "RUNNING" = Jobs, die aktiv sind.
Objekttyp	Kurzbezeichnung des Objekttyps oder "*" für alle Objekttypen. Format: AE-Name, Script-Literal oder Script-Variable
Bei den folgenden Parametern handelt es sich um optionale Filterkriterien. Sie können einen, zwei oder alle drei in beliebiger Kombination verwenden. Beachten Sie hierbei, dass die Kommata von übersprungenen Parametern trotzdem gesetzt werden müssen. Beispiel: `:SET &ANZAHL# = SYS_ACTIVE_COUNT("*", JOBS,,"MAWI.GRUPPE")`	
Objekt	Name eines Objekts oder Filter für mehrere Objekte. Format: AE-Name, Script-Literal oder Script-Variable Die Verwendung von Wildcardzeichen ist möglich. "*" steht dabei für beliebige, "?" für genau ein Zeichen.
Gruppe	Name einer Gruppe oder "*" für alle Gruppen Format: AE-Name, Script-Literal oder Script-Variable
Host	Name eines Agenten oder Filter für mehrere Agenten. Format: AE-Name, Script-Literal oder Script-Variable Die Verwendung von Wildcardzeichen ist möglich. "*" steht dabei für beliebige, "?" für genau ein Zeichen.

Rückgabewert
Anzahl der in den Aktivitäten stehenden Objekte.

Anmerkungen

Um zu überprüfen wie viele Objekte sich im Aktivitätsfenster befinden, kann die Script-Funktion SYS_ACTIVE_COUNT verwendet werden. Mit Parametern können Sie die Ermittlung auf bestimmte Aufgaben einschränken. Die Filterung auf Status, Objekttyp, Objektname, Gruppe sowie Host ist in beliebiger Kombination möglich.

Der Parameter *Host* ermöglicht es unter anderem auch die Parallelität eines Jobs auf einem bestimmten Rechner zu steuern. Wird ein Job in regelmäßigen Abständen ausgeführt, ist eine Überschneidung ausschließbar, wenn Sie zuvor mit SYS_ACTIVE_COUNT prüfen, ob dieser noch aktiv ist. Der Zielhost kann nämlich auch per Script-Anweisung :PUT_ATT nachträglich gesetzt werden. Bei FileTransfers werden sowohl Quell- als auch Zielhost miteinbezogen.

⚠ Die Verwendung des Parameters Host ist natürlich nur dann sinnvoll, wenn Sie beim Objekttyp "JOBS", "JOBF", "EVNT" oder "*" angegeben haben. Die Filterung nach Hosts schließt alle anderen Objekttypen aus, da diese keinen Agenten für die Durchführung benötigen. Das folgende Scriptbeispiel liefert daher immer "0".

```
:SET &ANZAHL# = SYS_ACTIVE_COUNT(ANY_ABEND,JOBP,,,"*")
```

⚠ Beachten Sie, dass nicht überprüft wird, ob der angegebene *Host* existiert.

Bei der Filterung nach aktiven Jobs mit dem Parameter "RUNNING", werden alle jene berücksichtigt, die einen System-Rückgabewert im Bereich von 1500 bis 1599 aufweisen.

Beachten Sie, dass ab der Automation Engine-Version 6.00A die Performance der Script-Funktion erhöht worden ist. Die Datenbanken MS SQL Server und DB2 profitieren davon. Die Performancesteigerung macht sich vor allem dann bemerkbar, wenn viele SYS_ACTIVE_COUNTs hintereinander aufgerufen werden. Für die Aufgabenzählung wird nur mehr eine Momentaufnahme erstellt. Dies hat zur Folge, dass auch Aufgaben mitgezählt werden, die noch nicht in der Datenbank bestätigt worden sind. Technisch wird diese Methode auch als "uncommited read" bezeichnet.

Beispiele

Die Beispiele zählen die in den Aktivitäten stehenden Objekte. Dabei wird die Anzahl aller Blockierer, die Anzahl aller abgebrochenen Objekte und die Anzahl aller Ereignisse ermittelt.

```
:SET &ANZAHL# = SYS_ACTIVE_COUNT("BLOCKED", "*")
:SET &ANZAHL# = SYS_ACTIVE_COUNT(ANY_ABEND, "*")
:SET &ANZAHL# = SYS_ACTIVE_COUNT("*", EVNT)
```

Das Beispiel zählt Aufgaben, welche für eine Gruppe vorgemerkt sind. Im Namen der Aufgaben muss der String "FILE" vorkommen.

```
:SET &ANZAHL# = SYS_ACTIVE_COUNT("PREPARED", "*", "*FILE*","GRP7")
```

Das nächste Beispiel ermittelt die Jobanzahl, die den Agenten "UNIX01" eingetragen haben.

```
:SET &ANZAHL# = SYS_ACTIVE_COUNT("*", "JOBS", "*",,"UNIX01")
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_STATE_JOBS_IN_GROUP	Ermittelt die Anzahl von Jobs, die in Gruppen vorgemerkt sind.

Script-Sprachmittel - Objekte aktivieren
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.3.16 SYS_STATE_JOB_ACTIVE

Script-Funktion: Prüft, ob ein Job bereits aktiv ist.

Syntax

SYS_STATE_JOB_ACTIVE(*[Job]*)

Syntaxteil	Beschreibung/Format
Job	Name eines Jobs Format: AE-name, Script-Literal oder Script-Variable

Rückgabewerte
"Y" - Der Job ist in einem Status mit System-Returncode < 1699 oder gleich 1701. "N" - Der Job ist in einem Status mit System-Returncode > 1699 und ungleich 1701.

Anmerkungen

Die Script-Funktion prüft, ob sich der angegebene Job in einem Status mit System-Returncode kleiner 1699 befindet oder exakt dem Returncode 1701 entspricht. Wird der Parameter nicht angegeben, kontrolliert die Funktion, das Objekt in dem sie aufgerufen wird.

Die Prüfung erfolgt genau zu dem Zeitpunkt, an dem die Script-Zeile, welche die Funktion enthält, abgearbeitet wird. Ein negatives Ergebnis sagt nichts darüber aus, dass das zu prüfende Objekt nicht später aktiviert werden kann. Mit Hilfe der Funktionen kann daher keine Synchronisation von Abläufen erfolgen. Sie liefert nur eine zeitlich begrenzte Information über den Status von Jobs.

Wird der Name des eigenen Objektes angegeben (= Objekt, über welches das Sprachmittel ausgeführt wird), so wird dies auch berücksichtigt und immer der Rückgabewert "Y" zurückgeliefert.

Beispiel

Im Beispiel wird ermittelt, ob ein Job momentan aktiv ist, und das Ergebnis an eine Script-Variable übergeben.

```
:SET  &AKTIV# = SYS_STATE_JOB_ACTIVE(MAWI.ABSCHLUSS)
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_STATE_ACTIVE	Prüft, ob ein Objekt bereits aktiv ist.
SYS_STATE_JOBS_IN_GROUP	Ermittelt die Anzahl von Jobs, die in Gruppen vorgemerkt sind.
SYS_STATE_JP_ACTIVE	Prüft, ob ein Workflow bereits aktiv ist.

Script-Sprachmittel - Objekte aktivieren
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.3.17 SYS_STATE_JOBS_IN_GROUP

Script-Funktion: Ermittelt die Anzahl von Jobs, die in Gruppen vorgemerkt sind.

Syntax

SYS_STATE_JOBS_IN_GROUP*([Job[, Gruppe]])*

Syntaxteil	Beschreibung/Format
Job	Name eines Jobs Format: AE-name, Script-Literal oder Script-Variable
Gruppe	Name einer Gruppe Format: AE-Name, Script-Literal oder Script-Variable

Rückgabewert
Anzahl an Jobs, die für Gruppen vorgemerkt sind.

Anmerkungen

Die Funktion ermittelt die Anzahl von Jobs, die den Status "Vorgemerkt" besitzen. Mit den Parameter können Sie gezielt einen bestimmten Job oder eine Gruppe angeben für die die Anzahl erhoben wird.

Beispiele

Das erste Beispiel ermittelt alle vorgemerkten Jobs.

```
:SET  &AKTIV# = SYS_STATE_JOBS_IN_GROUP()
```

Das zweite Beispiel liefert alle Jobs, die für die Gruppe "MAWI.GRUPPE" vorgemerkt sind.

```
:SET  &AKTIV# = SYS_STATE_JOBS_IN_GROUP(,"MAWI.GRUPPE")
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_STATE_ACTIVE	Prüft, ob ein Objekt bereits aktiv ist.
SYS_STATE_JOB_ACTIVE	Prüft, ob ein Job bereits aktiv ist.
SYS_STATE_JP_ACTIVE	Prüft, ob ein Workflow bereits aktiv ist.

Script-Sprachmittel - Objekte aktivieren
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.3.18 SYS_STATE_JP_ACTIVE

Script-Funktion: Prüft, ob ein Workflow bereits aktiv ist.

Syntax

SYS_STATE_JP_ACTIVE(*[Workflow]***)**

Syntaxteil	Beschreibung/Format
Workflow	Name eines Workflows Format: AE-name, Script-Literal oder Script-Variable

Rückgabewerte
"Y" - Der Workflow ist in einem Status mit System-Returncode < 1699 oder gleich 1701. "N" - Der Workflow ist in einem Status mit System-Returncode > 1699 und ungleich 1701.

Anmerkungen

Die Script-Funktion prüft, ob sich der angegebene Workflow in einem Status mit System-Returncode kleiner 1699 befindet oder exakt dem Returncode 1701 entspricht. Wird der Parameter nicht angegeben, kontrolliert die Funktion das Objekt in dem sie aufgerufen wird.

Die Prüfung erfolgt genau zu dem Zeitpunkt an dem die Script-Zeile, welche die Funktion enthält, abgearbeitet wird. Ein negatives Ergebnis sagt nichts darüber aus, dass das zu prüfende Objekt nicht später aktiviert werden kann. Mit Hilfe der Funktionen kann daher keine Synchronisation von Abläufen erfolgen. Sie liefert nur eine zeitlich begrenzte Information über den Status von Workflows.

Wird der Name des eigenen Objektes angegeben (= Objekt, über welches das Sprachmittel ausgeführt wird), so wird dies auch berücksichtigt und immer der Rückgabewert "Y" zurückgeliefert.

Beispiele

Das Beispiel prüft, ob der Workflow "FIBU.TAGESABSCHLUSS" aktiv ist. Der Name des Workflows wird mit einer Script-Variablen übergeben.

```
:SET  &JP# = "FIBU.TAGESABSCHLUSS"
:SET  &AKTIV# = SYS_STATE_JP_ACTIVE(&JP#)
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_STATE_ACTIVE	Prüft, ob ein Objekt bereits aktiv ist.
SYS_STATE_JOB_ACTIVE	Prüft, ob ein Job bereits aktiv ist.
SYS_STATE_JOBS_IN_GROUP	Ermittelt die Anzahl von Jobs, die in Gruppen vorgemerkt sind.

Script-Sprachmittel - Objekte aktivieren

Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.3.19 TOGGLE_OBJECT_STATUS

Script-Funktion: Stoppt oder startet die automatische Verarbeitung von manchen Objekttypen.

Syntax

TOGGLE_OBJECT_STATUS*(RunID, Status [,Aufgaben])*

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) des aktivierten Objektes Format: AE-Name, Script-Literal oder Script-Variable
Status	Verarbeitungsstatus, der gesetzt werden soll: Format: AE-Name, Script-Literal oder Script-Variable Erlaubte Werte: "STOP" und "GO" "STOP" - Stoppt die automatische Verarbeitung von Aufgaben. "GO" - Startet die automatische Verarbeitung von Aufgaben.
Aufgaben	ALL - Es wird auch die automatische Verarbeitung der Aufgaben gestoppt oder gestartet, die innerhalb von Workflows, Gruppen und Schedules laufen. Format: AE-Name, Script-Literal oder Script-Variable

Rückgabewert
"0" - Die Statusänderung war erfolgreich.

Anmerkungen

Die Script-Funktion ändert die Verarbeitung von Workflows, Gruppen, Ereignissen, RemoteTaskManager und Schedules.

Zur Ausführung der Script-Funktion ist die Berechtigung "Modifizieren zur Laufzeit" notwendig!

Beispiel

Im Beispiel wird die laufende Nummer (RunID) eines Ereignisses ermittelt und in eine Script-Variable eingelesen. Die automatische Verarbeitung des Ereignisses wird gestoppt. Dabei wird die ermittelte laufende Nummer (RunID) mit der Script-Variablen übergeben.

```
:SET  &RUNNR# = GET_UC_OBJECT_NR(DUMP.CONTROL)
:SET  &RET# = TOGGLE_OBJECT_STATUS(&RUNNR#, "STOP")
:PRINT  &RET#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
TOGGLE_SYSTEM_STATUS	Stoppt oder startet die automatische Verarbeitung eines gesamten Mandanten.

Script-Sprachmittel - Objekte aktivieren
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4 Objekte lesen oder verändern

3.4.1 :ADD_ATT

Script-Anweisung: Fügt einem Benachrichtigungs-Objekt Empfänger zur Laufzeit hinzu.

Syntax

:ADD_ATT RECIPIENT ***, Empfänger*** *[, Kalender, Kalenderbegriff]*

Syntaxteil	Beschreibung/Format
RECIPIENT	Name des Attributes, das hinzugefügt werden soll. Format: AE-name, Script-Literal oder Script-Variable Erlaubter Wert: "RECIPIENT"
Empfänger	Name eines Benutzers, einer BenutzerGruppe oder eine E-Mail-Adresse Format: Script-Literal oder Script-Variable
Kalender	Name eines Kalender-Objektes. Format: Script-Literal oder Script-Variable
Kalenderbegriff	Kalenderbegriff innerhalb dieses Kalenders. Format: Script-Literal oder Script-Variable

Anmerkungen

Diese Script-Anweisung kann nur in der Script-Registerkarte eines Benachrichtigungs-Objektes eingesetzt werden. Sie fügt den angegebenen Benutzer, die BenutzerGruppe oder E-Mail-Adresse der Liste der zu verständigenden Empfänger hinzu. Dabei kann auch ein Kalender definiert werden. Der Empfänger wird hierbei auch nur dann verständigt, wenn der Kalenderbegriff zutrifft.

Bereits eingetragene Empfänger werden dadurch nicht überschrieben, da die Script-Anweisung das Benachrichtigungs-Objekt nicht ändert. Die Ergänzung der Empfänger liste gilt nur für dessen Durchführung.

Im Report ist ersichtlich welche Empfänger die Script-Anweisung hinzugefügt hat.

Beachten Sie, dass der Benutzer im Format *Benutzername/Abteilung* angegeben werden muss. Dies entspricht dem Namen des Benutzer-Objekts.

Beispiele

Im Beispiel wird der Benutzer "BU/UC4" der Liste der zuständigen Empfänger hinzugefügt. Er wird aber nur benachrichtigt, wenn der Kalenderbegriff zutrifft.

```
:ADD_ATT RECIPIENT, "BU/UC4", "BEREITSCHAFT","WOCHENTAGS"
```

Das zweite Beispiel informiert alle Mitglieder der BenutzerGruppe "ADMIN".

```
:ADD_ATT RECIPIENT, "ADMIN"
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:REMOVE_ATT	Entfernt in einem Benachrichtigungs-Objekt Empfänger zur Laufzeit.
:PUT_ATT	Setzt oder ändert Attribute von Objekten.
:PUT_ATT_APPEND	Erweitert den Meldungstext einer Benachrichtigung zur Laufzeit.
GET_ATT	Liefert den Wert von Attributen einer Aufgabe während der Generierung.
GET_ATT_SUBSTR	Liefert einen Teil des Meldungstextes einer Benachrichtigung.

Script-Sprachmittel - Objekte lesen oder verändern
Beispielsammlung
Benachrichtigung mit variablem Meldungstext
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.2 :ADD_COMMENT

Script-Anweisung: Fügt einer Aufgabe einen Kommentar hinzu.

Syntax

:ADD_COMMENT *[RunID]* ***, Kommentar***

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) der Aufgabe der ein Kommentar hinzugefügt werden soll. Format: Script-Literal, Script-Variable oder Zahl Geben Sie keine *RunID* an, so wird der Kommentar bei jener Aufgabe hinterlegt, welche die Script-Anweisung aufruft.
Kommentar	Text des Kommentars Format: Script-Literal oder Script-Variable

Anmerkungen

Der Kommentareintrag enthält auch den Namen eines Benutzer-Objektes. Der Benutzer, der für die Aktivierung des Objektes verantwortlich ist, das die Script-Anweisung aufruft, wird automatisch eingetragen.

Der Umfang des Kommentars ist insofern begrenzt als dass eine Scriptzeile grundsätzlich nicht länger als 1024 Zeichen sein darf.

Beispiele

Das Beispiel fügt für die Aufgabe einen Kommentar hinzu, der in der Script-Variable &TEXT# enthalten ist.

```
:ADD_COMMENT ,&TEXT#
```

Im zweiten Beispiel wird der Kommentar beim obersten Workflow hinterlegt.

```
:SET  &TOP_JP_RUNID# = SYS_ACT_TOP_NR()
:ADD_COMMENT  &TOP_JP_RUNID#, "Neustart des FileTransfers war
erfolgreich!"
```

Siehe auch:

Script-Sprachmittel	Beschreibung
PREP_PROCESS_ COMMENTS	Bereitet die Bearbeitung einer Datensequenz (Kommentare einer Aufgabe) vor

Script-Sprachmittel - Objekte lesen oder verändern
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.3 :ATTACH_SYNC

Script-Anweisung: Weist einer Aufgabe ein Sync-Objekt zu.

Syntax

:ATTACH_SYNC *[Objekttyp,] [RunID],* ***Sync-Objekt,*** *[Start-Aktion], [Abend-Aktion], [End-Aktion],* ***Sonst,*** [**NEXT_OBJECT**]

Syntaxteil	Beschreibung/Format

Objekttyp	Kurzbezeichnung eines aktivierbaren Objektes Format: Script-Literal oder Script-Variable
RunID	Laufende Nummer (RunID) des Objektes. Format: Script-Literal, Script-Variable oder Zahl
Sync-Objekt	Name eines Sync-Objektes, mit dem die Aufgabe synchronisiert werden soll. Format: Script-Literal oder Script-Variable
Start-Aktion	Aktion, die beim Start der Aufgabe durchgeführt werden soll. Format: Script-Literal oder Script-Variable
Abend-Aktion	Aktion, die beim Abbruch der Aufgabe durchgeführt werden soll. Format: Script-Literal oder Script-Variable
End-Aktion	Aktion, die beim Beenden der Aufgabe durchgeführt werden soll. Format: Script-Literal oder Script-Variable
Sonst	Behandlung der Aufgabe, wenn weder Start-Aktion, Abend-Aktion noch End-Aktion ausgeführt werden können. Format: Script-Literal oder Script-Variable Erlaubte Werte: "A", "W" oder "S" "A" - Abbrechen (Abend) "W" - Warten (Wait) "S" - Überspringen (Skip)
NEXT_OBJECT	Das Sync-Objekt soll dem direkten Nachfolger in einem Workflow zugewiesen werden. Format: Script-Literal oder Script-Variable

Anmerkungen

Die Script-Anweisung ermöglicht es, einer bereits aktivierten Aufgabe ein Sync-Objekt hinzuzufügen. Die Zuweisung erfolgt temporär, d.h. nur für die Dauer ihrer Durchführung. *Objekttyp* ist ein optionaler Parameter, da der Objekttyp durch die laufende Nummer (RunID) eindeutig zugeordnet werden kann.

Die Script-Anweisung kann dazu verwendet werden, um beispielsweise in einem Workflow einer der nachfolgenden Aufgaben ein Sync zuzuweisen, und so die weitere Verarbeitung zu beeinflussen. Bei der Aufgabe, deren Script die Script-Anweisung enthält, muss die Option **Zur Laufzeit generieren** gesetzt sein. Die nachfolgende Aufgabe, der ein Sync zugewiesen werden soll, muss bereits aktiviert worden sein und damit eine RunID besitzen. Sie darf daher nicht die Option **Zur Laufzeit generieren** gesetzt haben. Mit dem Parameter NEXT_OBJECT kann dem direkten Nachfolger einer Aufgabe in einem Workflow ein Sync zugewiesen werden. Als Einschränkung gilt dabei, dass nur ein Nachfolger existieren darf.

Mit der Script-Anweisung kann auch der eigenen Aufgabe ein Sync zugewiesen werden. Die Sync-Bedingungen könnten dabei beispielsweise variabel gebildet werden. Der Parameter *RunID* muss hier nicht angegeben werden. Es ist jedoch das Komma für diesen nicht verwendeten Parameter zu setzen. Die Option **Zur Laufzeit generieren** darf in diesem Fall nicht aktiviert sein, da die Prüfung auf Sync-Objekte zu dem Zeitpunkt, an dem das Script durchgeführt wird, bereits abgeschlossen ist.

⚠ Das Script-Sprachmittel besitzt keine Wirkung auf Script-Objekte, da diese Objekte mit der Generierung bereits abgeschlossen sind, die Synchronisation jedoch erst danach erfolgt.

Soll einer anderen, bereits aktivierten Aufgabe ein Sync zugewiesen werden, ist das generell nur bei bestimmten Status möglich. Die Aufgabe darf entweder noch nicht gestartet sein (System-Rückgabewert < 1540) oder sie befindet sich in einem Wartezustand (System-Rückgabewert zwischen 1600 und 1700). Hat die Aufgabe einen anderen Zustand, führt dies zu einem Laufzeitfehler und zum Abbruch des Scripts.

Beispiele

Im ersten Beispiel wird dem Job "ARCHIV01" das Sync-Objekt "SYSTEM_0001_EXCLUSIVE_SYNC" hinzugefügt. Sollten die angegebenen Aktionen nicht ausgeführt werden können, wird der Job abgebrochen.

```
:SET &RUNID# = GET_UC_OBJECT_NR(ARCHIV01)
:ATTACH_SYNC &RUNID#,"SYSTEM_0001_EXCLUSIVE_
SYNC","USE","RELEASE","RELEASE","A"
```

Im zweiten Beispiel wird das Sync-Objekt der direkt nachfolgenden Aufgabe im Workflow zugewiesen.

```
:ATTACH_SYNC ,"SYSTEM_0001_
EXCLUSIVE.SYNC","USE","RELEASE","RELEASE","A","NEXT_OBJECT"
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SET_SYNC	Führt die definierte Aktion eines Sync-Objektes aus.
GET_SYNC	Fragt den aktuellen Zustand oder Wert eines Sync-Objektes ab.

Script-Sprachmittel - Objekte lesen oder verändern
Sync
System-Rückgabewerte von aktivierbaren Objekten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.4 :DELETE_VAR

Script-Anweisung: Löscht einen oder alle Werte aus einem statischen Variablen-Objekt.

Syntax

:DELETE_VAR ***Variable*** *[,Key]*

Syntaxteil	Beschreibung/Format
Variable	Name des Variablen-Objektes, in der Inhalte gelöscht werden sollen. Format: AE-name, Script-Literal oder Script-Variable
Key	Key, der samt Wert gelöscht werden soll. Format: AE-Name, Script-Literal oder Script-Variable

Anmerkungen

⚠ Beachten Sie, dass dieses Scriptsprachmittel nur für statische Variablen-Objekte angewandt werden kann. Für mehr Informationen siehe Beschreibung der Registerkarte **Variable**.

Die Script-Anweisung löscht alle Zeilen in einem Variablen-Objekt, wenn Sie nur den Parameter *Variable* angeben.

Verwenden Sie zusätzlich den Parameter *Key*, so wird nur die entsprechende Zeile aus dem Variablen-Objekt entfernt. Geben Sie beim Key ein Sternchen "*" an, wird hingegen der gesamte Inhalt des Variablen-Objektes gelöscht.

⚠ Beachten Sie, dass die Verwendung von Wildcardzeichen nicht erlaubt ist! Sie können daher mit einem "*" oder "?" im Parameter *Key* nicht auf mehrere Zeilen filtern!

Beispiele

Im ersten Beispiel löscht die Script-Anweisung den ganzen Inhalt.

```
:DELETE_VAR "DATENBANKWARTUNG"
```

Das zweite Beispiel löscht nur die Zeile, die den Key "Mandant" besitzt.

```
:DELETE_VAR "DATENBANKWARTUNG", "Mandant"
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:PUT_VAR	Speichert einen Wert in ein statisches Variablen-Objekt.
GET_VAR	Liefert den Inhaltseintrag eines Variablen-Objektes.
:SET_SCRIPT_VAR	Setzt die Werte von Script-Variablen per indirektem Zugriff.
GET_SCRIPT_VAR	Liefert die Werte von Script-Variablen per indirektem Zugriff.
PREP_PROCESS_VAR	Bereitet die Bearbeitung einer Datensequenz (Werte eines Variablen-Objektes) vor.

Script-Sprachmittel - Objekte lesen oder verändern
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.5 :MODIFY_STATE

Script-Anweisung: Ändert den Rückgabewert oder Statustext eines Jobs nach dessen Ende.

Syntax

:MODIFY_STATE*Eigenschaft=Wert*

Syntaxteil	Beschreibung/Format

Eigenschaft	Eigenschaft für das Jobende. Format: Script-Variable oder AE-name Erlaubte Werte: "RETCODE", "STATUS_TEXT" "RETCODE" = Rückgabewert des Jobs. "STATUS_TEXT" = Statustext des Jobs.
Wert	Neue Zuweisung für die Eigenschaft des Job-Endes. Format: Script-Literal, Script-Variable, AE-Name, Zahl ohne Hochkommata oder Script-Funktion Für "RETCODE": Numerischer Wert. Für "STATUS_TEXT": Alphanumerische Zeichenfolge, maximal 32 Zeichen.

Anmerkungen

Mit :MODIFY_STATE kann der Rückgabewert oder der Statustext eines Jobs nachträglich geändert werden. Die Script-Anweisung ist daher nur in der Registerkarte **Post-Script** erlaubt.

Die Script-Anweisung kann beispielsweise dazu verwendet werden, um einem Job, der "technisch" zwar normal endete, nachträglich den Status ENDED_NOT_OK zuzuweisen. Das könnte bei einem Job notwendig werden, bei dem ein Fehler erst durch die Analyse des eigenen Reports erkannt werden kann (PREP_PROCESS_REPORT).

Die Änderung des Rückgabewertes beeinflusst den Status des Jobs. Der Status resultiert daraus, welcher maximale Rückgabewert für ein normales Ende des Jobs definiert wurde (Registerkarte **Laufzeit**). Ist der gesetzte Rückgabewert größer als dieser maximale Rückgabewert wird der Status auf ENDED_NOT_OK gesetzt. Ist der gesetzte Rückgabewert kleiner als der festgelegte maximale Rückgabewert oder gleich, erhält der Job den Status ENDED_OK.

Für das Jobende kann auch ein eigener Statustext gesetzt werden. Er ersetzt den Text, den der Jobmelder im Trailer ausgegeben hat.

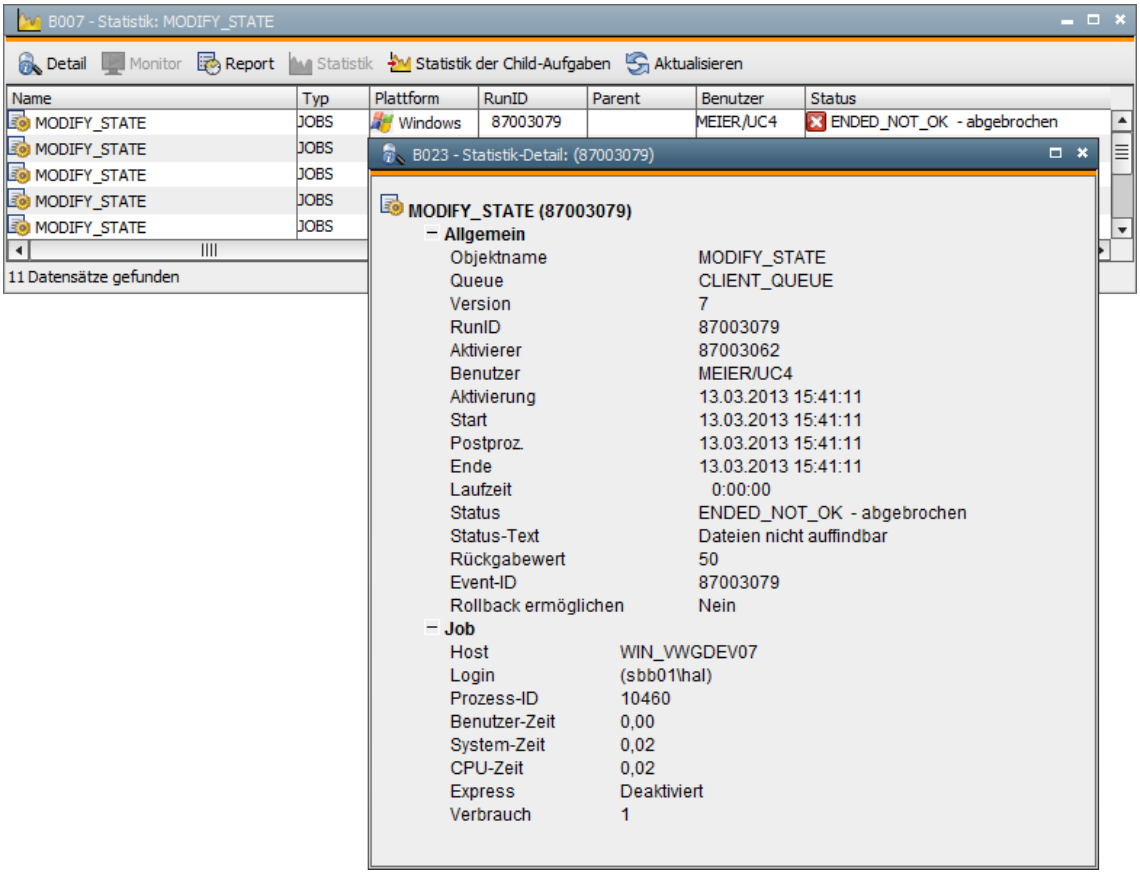
Die Änderungen werden im Report des Jobs, Registerkarte "Post-Prozessierung", protokolliert. Die geänderten Werte sind auch im Detailfenster des Jobs sichtbar.

Beispiel

Im Beispiel geht es um einen Job, der unter Windows eine Datei kopieren soll, die nicht vorhanden ist. Der Job würde normal mit Rückgabewert "0" enden. Nur aus dem Job-Report wäre ersichtlich, dass die zu kopierende Datei nicht gefunden wurde.

Im Post-Script des Jobs wird nun der Job-Report analysiert. Das Fehlen der zu kopierenden Datei wird erkannt und der Rückgabewert geändert. Dadurch wird der Job abgebrochen.

```
:SET  &HND# = PREP_PROCESS_REPORT(,,,"*Datei nicht finden*")
:PROCESS  &HND#
:  MODIFY_STATE STATUS_TEXT="Dateien nicht auffindbar"
:  MODIFY_STATE RETCODE=50
:ENDPROCESS
```



Siehe auch:

Script-Sprachmittel	Beschreibung
:EXIT	Beendet die Prozessierung des Scripts mit Rückgabewert.
GET_UC_OBJECT_STATUS	Liefert den Status eines aktivierten Objektes.

Script-Sprachmittel - Objekte lesen oder verändern
Beispielsammlung
Endestatus aufgrund des Reportinhaltes setzen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.6 :PUT_ATT

Script-Anweisung: Ändert den Wert eines Attributes während der Generierung.

Syntax

:PUT_ATT *Attribut = Wert*

Syntaxteil	Beschreibung/Format

Attribut	Name des Attributes, welchem ein Wert zugewiesen werden soll. Format: AE-name
Wert	Passender Wert, der zu dem Attribut zugewiesen werden soll. Format: Script-Literal, Script-Variable oder Script-Funktion

Anmerkungen

Standardmäßig werden Objekte mit den Attributen ausgeführt, mit denen sie definiert wurden. Mit der Anweisung :PUT_ATT erhalten Sie die Möglichkeit, ein *Attribut* dynamisch während der Generierung zu ändern, ihm einen neuen *Wert* zuzuweisen. Dieser *Wert* ist für die jeweilige Generierung gültig, wird aber nicht dauerhaft im Objekt gespeichert.

Der *Wert*, welcher einem *Attribut* zugewiesen werden kann, ist sehr eng mit der Systemumgebung verbunden. So kann beispielsweise bei einem Job als Starttyp nur " " für einen sofortigen Start oder eine bereits definierte Gruppe zugewiesen werden. Andere Angaben würden zu einem Fehler bei der Generierung führen.

Im Benutzerhandbuch finden Sie eine Liste der Attribute aller Objekte mit ihren möglichen Werten.

Die Attribute von Rapid Automation Jobs können sich mit der Freigabe einer neuen Version eines RA-Agenten ändern. In diesem Fall funktionieren die :PUT_ATT-Befehle aus früheren Agenten-Versionen nicht mehr. Es ist auch möglich Objektvariablen anstelle von :PUT_ATT-Befehlen für Feldersetzungen zu verwenden. Gehen Sie dazu folgendermaßen vor:

1. Entfernen Sie den :PUT_ATT-Befehl aus der Job-Registerkarte Script oder Pre-Script.
2. Fügen Sie eine Variable plus Wert zur Job-Registerkarte Variablen & Prompts hinzu.
3. In das Feld, in dem der Wert ersetzt werden soll, tragen Sie nun die Objektvariable im Format &<Variable># ein. <Variable> steht für den Variablennamen, den Sie in der Registerkarte Variablen & Prompts definiert haben.

Beachten Sie, dass ab Version 3 des RA FTP- und Webservice-Agenten keine Feldersetzungen mehr für RA-Attribute vorgenommen werden! In diesem Fall sind die Attribute über :PUT_ATT zu setzen!

Das Ändern der Attribute ist natürlich nur vor der Durchführung des Objektes sinnvoll. Daher kann :PUT_ATT nicht in den Post-Script-Registerkarten verwendet werden.

Beachten Sie, dass die Attribute verschiedene maximale Längen besitzen! :PUT_ATT prüft nicht, ob der gesetzte Wert länger ist als erlaubt. Er wird auf die maximale Länge abgeschnitten.

Ist im Objekt die Option "Generieren zur Laufzeit" aktiviert, so können Sie den Starttyp nicht mit :PUT_ATT verändern!

Die Auflösung der AgentenGruppe und damit die Wahl auf welchem Agenten die Aufgabe läuft, erfolgt vor der gesamten Script-Durchführung! Sie können daher mit :PUT_ATT einen anderen Agenten setzen, aber keine AgentenGruppe! Sie haben jedoch die Möglichkeit mit der Script-Funktion PREP_PROCESS_AGENTGROUP die Agenten einer AgentenGruppe auszulesen.

In der Registerkarte **!Script** des Ereignis-Objektes hat die Script-Anweisung :PUT_ATT keine Auswirkung!

Die Attribute ATTR_DLG, HOST und bei BS2000-Jobs SYSLST_DB, SYSLST_FILE, SYSOUT_DB, SYSOUT_FILE, OL, OO können nur im Pre-Script mit :PUT_ATT gesetzt werden!

:PUT_ATT ändert nicht den Inhalt von Script-Variablen!

⚠ Verwenden Sie das Sprachmittel in der Script-Registerkarte, um die max. parallelen Durchführungen einzuschränken (Attribut: MAX_PARALLEL_TASKS), so wird diese Begrenzung erst nach der Scriptdurchführung geprüft, wenn die Aufgabe zur Laufzeit generiert wird.

Beispiel:

```
:SET&TYP# = GET_ATT(EVENT_TYPE)
:PUT_ATT EVENT_CHECK_METHOD2 = "FILE_STABLE"
:PRINT &TYP#
```

Angenommen die Script-Variable &TYP# erhält durch GET_ATT den Wert "FA". Nach dem :PUT_ATT-Aufruf ist der Typ des Ereignisses "FT", weil das Attribut "FILE_STABLE" zum Dateisystem-Ereignis gehört. Trotzdem enthält &TYP# nach wie vor den Wert "FA"!

⚠ Die Änderung von Attributen (zum Beispiel: Login-Objekt) eines bestimmten Objektes mit :PUT_ATT ist nicht möglich, wenn der Benutzer Schreibrecht (W) auf den alten Wert und kein Schreibrecht auf den neuen Wert des Attributes hat.

⚠ Um die Queue für die Aufgabe zu setzen (Attribut QUEUE), ist das Sprachmittel :PUT_ATT im Pre-Script zu verwenden. Dabei ist allerdings folgendes zu beachten:
Ist bei der Aufgabe die Option **Zur Laufzeit generieren** gesetzt, wird die Queue-Prüfung (Prüfung der verfügbaren Slots und des Queue-Status) bereits vor dem Pre-Script durchgeführt. Die Aufgabe wird zwar in die geänderte Queue gestartet, jedoch die Prüfung mit der im Objekt eingetragenen Queue durchgeführt. Dadurch wird das Limit der neuen Queue nicht geprüft und die Aufgabe läuft eventuell zu früh los.
Ist im Objekt beispielsweise noch die CLIENT_QUEUE eingetragen, so wird die Aufgabe auf jeden Fall gestartet (sofern diese nicht gestoppt ist).

Beispiel

Das Beispiel zeigt die Zuweisung eines anderen Agenten für das Objekt.

```
:PUT_ATT HOST = "UNIX02"
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:ADD_ATT	Fügt einem Benachrichtigungs-Objekt Empfänger zur Laufzeit hinzu.
:REMOVE_ATT	Entfernt in einem Benachrichtigungs-Objekt Empfänger zur Laufzeit.
:PUT_ATT_APPEND	Erweitert den Meldungstext einer Benachrichtigung zur Laufzeit.
GET_ATT	Liefert den Wert von Attributen einer Aufgabe während der Generierung.
GET_ATT_SUBSTR	Liefert einen Teil des Meldungstexts einer Benachrichtigung.

Script-Sprachmittel - Objekte lesen oder verändern
Beispielsammlung
Benachrichtigung mit variablem Meldungstext
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.7 :PUT_ATT_APPEND

Script-Anweisung: Erweitert den Meldungstext einer Benachrichtigung zur Laufzeit.

Syntax

:PUT_ATT_APPEND CALL_TEXT = *Text*

Syntaxteil	Beschreibung/Format
CALL_TEXT	Name des Attributes, dessen Wert erweitert werden soll. Format: AE-name und Script-Variable Erlaubter Wert: "CALL_TEXT"
Text	Text, der dem bestehenden Meldungstext hinzugefügt werden soll. Format: Script-Literal, Zahl, Script-Variable oder Script-Funktion

Anmerkungen

Diese Script-Anweisung kann nur in der Script-Registerkarte eines Benachrichtigungs-Objektes eingesetzt werden. Sie fügt am Ende des Meldungstextes den angegebenen Text hinzu.

Der bestehende Meldungstext wird dadurch nicht geändert, da die Script-Anweisung das Benachrichtigungs-Objekt nicht ändert. Die Erweiterung des Meldungstextes gilt nur für dessen Durchführung.

Bei der Verwendung von Leerzeichen gelten folgende Besonderheiten (siehe auch Beispiele):

- Leerzeichen am Ende des hinzuzufügenden Textes werden abgeschnitten.
- Besteht der Text nur aus Leerzeichen werden diese gänzlich ignoriert.

⚠ Beachten Sie, dass die Länge des übergebenen Textes aus Datenbank-technischen Gründen eine Länge von 8.000 Zeichen nicht überschreiten darf.

Beispiele

Im Beispiel wird die aktuelle Uhrzeit in einer Script-Variablen gespeichert und in den READ-Buffer gestellt. Danach wird ein Benachrichtigungs-Objekt aktiviert.

```
:SET  &TIME# = SYS_TIME("HH:MM")
:PUT_READ_BUFFER TIME# = "&TIME#"
:SET  &CALL# = ACTIVATE_UC_OBJECT(CALL, BACKUP.END)
```

Die Benachrichtigungs-Aufgabe liest den Wert der Script-Variable aus dem READ-Buffer. Die aktuelle Uhrzeit wird an den Meldungstext der Benachrichtigung angehängt.

```
:READ  &TIME#,,
:PUT_ATT_APPEND CALL_TEXT = " &TIME# Uhr."
```

Die folgenden Beispiele sollen Ihnen die Behandlungsweise von Leerzeichen demonstrieren. Der Meldungtext der Benachrichtigung lautet "Bitte Überprüfung der".

1) Der Text wird durch das führende Leerzeichen korrekt ausgegeben. Beachten Sie, dass nicht automatisch ein Leerzeichen eingefügt wird.

```
:PUT_ATT_APPEND CALL_TEXT = " Serverprozesse starten"
```

Ausgabe:

```
Bitte Überprüfung der Serverprozesse starten
```

2) Beachten Sie auch, dass Leerzeichen am Ende des Textes abgeschnitten werden.

```
:PUT_ATT_APPEND CALL_TEXT = " Serverprozesse "
:PUT_ATT_APPEND CALL_TEXT = "starten"
```

Ausgabe:

```
Bitte Überprüfung der Serverprozessestarten
```

3) Besteht der Text nur aus Leerzeichen, wird er ebenfalls ignoriert.

```
:PUT_ATT_APPEND CALL_TEXT = " Serverprozesse"
:PUT_ATT_APPEND CALL_TEXT = "    "
:PUT_ATT_APPEND CALL_TEXT = "starten"
```

Ausgabe:

```
Bitte Überprüfung der Serverprozessestarten
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:ADD_ATT	Fügt einem Benachrichtigungs-Objekt Empfänger zur Laufzeit hinzu.
:REMOVE_ATT	Entfernt in einem Benachrichtigungs-Objekt Empfänger zur Laufzeit.
:PUT_ATT	Ändert den Wert eines Attributes während der Generierung.
GET_ATT	Liefert den Wert von Attributen einer Aufgabe während der Generierung.
GET_ATT_SUBSTR	Liefert einen Teil des Meldungstextes einer Benachrichtigung.

Script-Sprachmittel - Objekte lesen oder verändern
Beispielsammlung
Benachrichtigung mit variablem Meldungstext
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.8 :PUT_VAR

Script-Anweisung: Speichert Werte in ein statisches Variablen-Objekt.

Syntax

:PUT[_VAR]*Variable,* *[Key]****, Wert*** *[, Wert [, Wert [, Wert [, Wert]]]]*

Syntaxteil	Beschreibung/Format
Variable	Name des Variablen-Objektes, welcher ein oder mehrere Wert zugewiesen werden soll. Format: AE-name oder Script-Variable

Key	Zeile in der die Werte gespeichert werden soll. Format: Script-Literal, Script-Variable oder Script-Funktion
Wert	Einträge, die in die jeweiligen Werte-Spalten geschrieben werden sollen. Werte von nicht angegebenen Spalten werden gelöscht. Bestehende Werte werden überschrieben. Format: Script-Literal, Zahl ohne Hochkommata oder Script-Variable

Anmerkungen

⚠ Wird bei dem Script-Sprachmittel eine dynamische Variable (Quelle: SQL, SQL-intern, Multi oder Datentyp) angegeben, führt dies zu einem Laufzeitfehler. Nur statische Variablen-Objekte können mit :PUT_VAR befüllt werden.

Für das Hinterlegen von Werten in einem Variablen-Objekt gelten folgende Regeln:

- Besitzt die Variable noch keinen Wert, werden der *Key* und der *Wert* hinzugefügt.
- Besitzt die Variable den angegebenen *Key* so werden dessen Werte ersetzt.

Um mehrere Werte-Spalten einer Variable zu befüllen, sind die einzelnen Werte durch Kommata getrennt anzugeben. Bei statischen Variablen gibt es maximal 5 Werte-Spalten.

Sie müssen den Parameter *Key* nicht angeben, wenn in der Registerkarte "Attribute" die Einstellung "Gültigkeitsbereich" - "kein Key" ausgewählt wurde. Die Variable enthält in diesem Fall nur einen Wert. Beachten Sie, dass bei ausgelassenem *Key* trotzdem die entsprechenden Kommata in der Anweisung zu setzen sind.

ℹ Die einzelnen Werte können auch ohne Hochkommata angeben werden. Kommata dienen dabei als Trennzeichen.
Beispiel: Befüllung aller 5 Werte-Spalten

```
:PUT_VAR VARA.TEST, "KEY1", Wert1, Wert2, Wert3, Wert4, Wert5
```

⚠ Um einen Wert in eine Spalte zu schreiben, der ein oder mehrere Kommata enthält, ist dieser unter einzelne oder doppelte Hochkomma zu setzen.
Beispiel: In Wertespalte 1 wird "Wert1" geschrieben und in Spalte 2 "Wert2, Wert3, Wert4"

```
:PUT_VAR VARA.TEST, "KEY1", Wert1, "Wert2, Wert3, Wert4"
```

Um einen Wert in eine Variable zu speichern, die den Datentyp "Zeitstempel" aufweist, muss eines der folgenden Formate gewählt werden:

- "JJJJ-MM-TT HH:MM:SS"
- "JJMMTT HHMMSS"
- "JJJJMMTT HHMMSS"
- "JJJJMMTTHH24MISS"

Bei den Datentypen "Zeit", "Datum" und "Zahl" ist der Wert im eingestellten Ausgabeformat (Registerkarte **Attribute**) anzugeben.
Für den Datentyp "Zahl" ist folgender Hinweis zu beachten: Wird versucht eine Zahl mit mehr Nachkommastellen zu speichern, als im Ausgabeformat erlaubt ist, werden die überflüssigen Dezimalstellen abgeschnitten.

ℹ Die Script-Funktion schreibt Werte auch dann in das Variablen-Objekt, wenn jenes gerade von einem Benutzer geöffnet ist, damit die Verarbeitung nicht behindert wird. Sollte der Benutzer das Variablen-Objekt ändern und speichern wollen, erhält er eine Warnung, dass die Variable in der Zwischenzeit geändert wurde. Er kann dann entscheiden, ob er die trotzdem speichert oder seine Änderungen verwirft.

Beim Datentyp "Text" bleiben führende Leerzeichen im Inhalt bestehen. Leerzeichen am Ende des Strings werden jedoch abgeschnitten.

Bei der Verwendung von :PUT_VAR vor einer :READ-Anweisung gilt folgende Besonderheit: Wird bei :READ die Generierung des Scripts manuell über die Schaltfläche "Abbrechen" oder aufgrund ungültiger Vorgabewerte (siehe "Zur Laufzeit generieren") abgebrochen, so besitzt das Variablen-Objekt trotzdem die mit :PUT_VAR gesetzten Werte.

Beispiel

Im Beispiel werden das aktuelle Tagesdatum und die aktuelle Tageszeit ermittelt und in einem Variablen-Objekt vom Typ "Zeitstempel" gespeichert. Tagesdatum und -zeit werden der Script-Anweisung als Script-Variablen übergeben.

```
:SET &DATUM# = SYS_DATE("JJJJ-MM-TT")
:SET &ZEIT# = SYS_TIME("HH:MM:SS")
:PUT_VAR BUCHUNGSDATUM, , "&DATUM# &ZEIT#"
```

Die Variable "ATTRIBUTE", die im folgenden Beispiel verwendet wird, enthält den Gültigkeitsbereich "Frei gewählt". Daher ist der *Key* unbedingt anzugeben.

```
:SET &PRIO# = GET_ATT("AE_PRIORITY")
:PUT_VAR ATTRIBUTE, "AktuellePrioritaet", &PRIO#
```

In folgendem Beispiel wird Name, Objekttyp und Informationen zum Parent einer Aufgabe unter der RunID der Aufgabe im Variablen-Objekt "OBJEKT_STATISTIK" abgelegt.

```
:SET &PNAME# = SYS_ACT_PARENT_NAME()
 :SET &PNR# = SYS_ACT_PARENT_NR()
 :SET &PTYPE# = SYS_ACT_PARENT_TYPE()
 :SET &NAME# = SYS_ACT_ME_NAME()
 :SET &NR# = SYS _ACT_ME_NR()
 :SET &TYPE# = SYS_ACT_ME_TYPE()

:PUT_VAR OBJEKT_STATISTIK, "&NR#", "Objektname: &NAME#", "Objekttyp:
&TYPE#", "Parent-Name: &PNAME#", "Parent-RunID: &PNR#", "Parent-
Objekttyp: &PTYPE#"
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:PUT_VAR_COL	Speichert einen Wert in eine bestimmte Spalte eines statischen Variablen-Objektes.
:DELETE_VAR	Löscht einen oder alle Werte aus einem statischen Variablen-Objekt.
GET_VAR	Liefert den Inhaltseintrag eines Variablen-Objektes.
:SET_SCRIPT_VAR	Setzt die Werte von Script-Variablen per indirektem Zugriff.
GET_SCRIPT_VAR	Liefert die Werte von Script-Variablen per indirektem Zugriff.
PREP_PROCESS_VAR	Bereitet die Bearbeitung einer Datensequenz (Werte eines Variablen-Objektes) vor.

Script-Sprachmittel - Objekte lesen oder verändern
Variable
Beispielsammlung:
Darstellungen mittels Cockpit
Datenbankwartung mit Optionen
Aufruf eines MBeans

Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.9 :PUT_VAR_COL

Script-Anweisung: Speichert einen Wert in eine bestimmte Spalte eines statischen Variablen-Objektes.

Syntax

:PUT_VAR_COL*Variable,* [Key]*, Spalte, Wert*

Syntaxteil	Beschreibung/Format
Variable	Name des Variablen-Objektes, welcher ein Wert zugewiesen werden soll. Format: AE-name oder Script-Variable
Key	Zeile, in welcher der Wert gespeichert werden soll. Format: Script-Literal, Script-Variable oder Script-Funktion
Spalte	Nummer der Spalte, in die der Wert eingetragen wird. Format: Script-Literal, Script-Variable oder Zahl ohne Hochkommata Erlaubte Werte: "1" bis "5"
Wert	Wert der in die angegebene Zeile und Spalte der Variable geschrieben werden soll. Format: Script-Literal oder Script-Variable

Anmerkungen

Diese Script-Funktion trägt einen bestimmten Wert in der angegebenen Zeile und Spalte in ein Variablen-Objekt ein. Dabei werden im Gegensatz zum Script-Sprachmittel :PUT_VAR andere Felder nicht beeinflusst. Ist bereits ein Eintrag in der angegebenen Zeile/Spalte vorhanden, wird dieser Wert überschrieben.

⚠ Wird bei dem Script-Sprachmittel eine dynamische Variable (Quelle: SQL, SQL-intern, Multi oder Datentyp) angegeben, führt dies zu einem Laufzeitfehler. Nur statische Variablen-Objekte können mit :PUT_VAR_COL befüllt werden.

⚠ Beachten Sie bei der Angabe der Spaltennummer, dass Variablen-Objekte vom Typ "statisch" nur 5 Werte-Spalten besitzen.

ℹ Bei Angabe der Spaltennummer ist der Bereich von "1" (Werte-Spalte 1) bis "5" (Werte-Spalte 5) erlaubt. Der Wert der Key-Spalte kann durch das Script-Sprachmittel nicht verändert werden.

ℹ Der Parameter *Key* ist nur dann optional, wenn eine statische Variable mit der Einstellung "Gültigkeitsbereich" - "Kein Key" verwendet wird. In diesem Fall besitzt die Variable nur einen Key (*).

ℹ Ist der angegebene *Key* noch nicht vorhanden, wird der Eintrag neu erstellt.

Beispiel

In folgendem Beispiel wird der Name der übergeordneten Aufgabe (Workflow) und die eigene RunID ermittelt. Danach wird die RunID unter dem Namen des Workflows in die Spalte 3 des Variablen-Objektes "VARA.JOBP" geschrieben.

```
:SET &JOBP# = SYS_ACT_PARENT_NAME()
:SET &RUNID# = SYS _ACT_ME_NR()
:PUT_VAR_COL VARA.JOBP, &JOBP#, "3", &RUNID#
```

Siehe auch:

:PUT_VAR	Speichert Werte in ein statisches Variablen-Objekt.
:DELETE_VAR	Löscht einen oder alle Werte aus einem statischen Variablen-Objekt.
GET_VAR	Liefert den Inhaltseintrag eines Variablen-Objektes.
:SET_SCRIPT_VAR	Setzt die Werte von Script-Variablen per indirektem Zugriff.
GET_SCRIPT_VAR	Liefert die Werte von Script-Variablen per indirektem Zugriff.
PREP_PROCESS_VAR	Bereitet die Bearbeitung einer Datensequenz (Werte eines Variablen-Objektes) vor.

Script-Sprachmittel - Objekte lesen oder verändern
Variable
Beispielsammlung:
Darstellungen mittels Cockpit
Datenbankwartung mit Optionen
Aufruf eines MBeans
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.10 :REMOVE_ATT

Script-Anweisung: Entfernt in einem Benachrichtigungs-Objekt Empfänger zur Laufzeit.

Syntax

:REMOVE_ATT RECIPIENT*, Empfänger*

Syntaxteil	Beschreibung/Format
RECIPIENT	Name des Attributes, das entfernt werden soll. Format: AE-name, Script-Literal oder Script-Variable Erlaubter Wert: "RECIPIENT"
Empfänger	Name eines Benutzers, einer BenutzerGruppe oder eine E-Mail-Adresse Format: Script-Literal, Script-Variable oder Zahl Für Benutzer- und BenutzerGruppen-Objekte können Sie die Wildcardzeichen "*" und "?" verwenden. "*" steht dabei für beliebige, "?" für genau ein Zeichen.

Anmerkungen

Diese Script-Anweisung kann nur in der Script-Registerkarte eines Benachrichtigungs-Objektes eingesetzt werden. Sie entfernt die angegebenen Benutzer, BenutzerGruppen oder E-Mail-Adressen aus der Liste der zu verständigenden Empfänger.

Die eingetragenen Empfänger werden nicht dauerhaft gelöscht, da die Script-Anweisung das Benachrichtigungs-Objekt nicht ändert. Die Anpassung der Empfängerliste gilt nur für dessen Durchführung.

⚠ Bei der Verwendung von Wildcard-Zeichen ist darauf zu achten, dass dadurch nicht alle zuständigen Empfänger entfernt werden. Dies führt zum Abbruch der Script-Verarbeitung und zur Ausgabe einer Fehlermeldung.

ℹ Im Report ist ersichtlich welche Empfänger die Script-Anweisung entfernt hat.

⚠ Beachten Sie, dass der Benutzer im Format *Benutzername/Abteilung* angegeben werden muss. Dies entspricht dem Namen des Benutzer-Objekts. Wird der Benutzer nicht gefunden, bricht das Script ab.

Beispiele

Im Beispiel wird der Benutzer "BU/UC4" aus der Liste der zuständigen Empfänger entfernt.

```
:REMOVE_ATT RECIPIENT,"BU/UC4"
```

Das zweite Beispiel entfernt alle Benutzer der Abteilung "UC4".

```
:REMOVE_ATT RECIPIENT,"*/UC4"
```

Die BenutzerGruppe "ADMIN" wird im dritten Beispiel entfernt.

```
:REMOVE_ATT RECIPIENT,"ADMIN"
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:ADD_ATT	Fügt einem Benachrichtigungs-Objekt Empfänger zur Laufzeit hinzu.
:PUT_ATT	Ändert den Wert eines Attributes während der Generierung.
:PUT_ATT_APPEND	Erweitert den Meldungstext einer Benachrichtigung zur Laufzeit.
GET_ATT	Liefert den Wert von Attributen einer Aufgabe während der Generierung.
GET_ATT_SUBSTR	Liefert einen Teil des Meldungstextes einer Benachrichtigung.

Script-Sprachmittel - Objekte lesen oder verändern
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.11 :REPLACE_STRUCTURE

Script-Anweisung: Ersetzt die Struktur eines Workflows bei dessen Aktivierung durch die Struktur eines anderen Workflows.

Syntax

:REPLACE_STRUCTURE OBJ[ECT]=*Workflow*

Syntaxteil	Beschreibung/Format
Workflow	Name des Workflows, dessen Struktur verwendet werden soll. Format: Script-Literal oder Script-Variable

Anmerkungen

Mit der Script-Anweisung kann während der Aktivierung die Struktur des eigenen Workflows durch die Struktur eines anderen Workflows ersetzt werden. Dies bedeutet mit anderen Worten, dass der Inhalt der Registerkarte "Workflow" ausgetauscht wird. Es tritt ein Laufzeitfehler auf, wenn der angegebene Workflow nicht gefunden werden kann.

Die Veränderung der Struktur des Workflows gilt nur für die aktuelle Durchführung und verändert nicht das Workflow-Objekt.

Wenn Sie zu einer Workflow-Durchführung in der Statistik den Monitor aufrufen, wird der Workflow angezeigt, dessen Struktur verwendet worden ist.

Die Script-Anweisung sollte nicht verwendet werden, wenn die Modifizierungen beispielsweise auch durch die Definition von Kalenderabhängigkeiten, zeitlichen Bedingungen in Scripts u.a. zu realisieren sind.

Beispiel

Im Beispiel soll der Workflow "MAWI.TAG" einmalig am 31.12.2005 mit einer modifizierten Workflow-Struktur laufen. Im Original-Workflow wird der modifizierte Workflow mit den folgenden Script-Zeilen aufgerufen.

```
:IF  "051231" = SYS_LDATE()
:   REPLACE_STRUCTURE OBJECT="MAWI.TAG.MOD.051231"
:ENDIF
```

Siehe auch:

Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.12 :SET_CALE

Script-Anweisung: Fügt einem Kalenderbegriff ein Datum oder einen Zeitraum hinzu oder entfernt diese.

Syntax

:SET_CALE*Kalender, Kalenderbegriff, Datum1*[,Datum2[,Aktion]]

Syntaxteil	Beschreibung/Format
Kalender	Name des Kalender-Objektes, dem ein Datum/Zeitraum hinzugefügt bzw. entfernt werden soll. Format: AE-name, Script-Literal oder Script-Variable
Kalenderbegriff	Name des Kalenderbegriffs, dem ein Datum/Zeitraum hinzugefügt bzw. entfernt werden soll. Format: AE-Name, Script-Literal oder Script-Variable
Datum1 *Datum2*	Angabe eines Datums im Format "JJMMTT" oder "JJJJMMTT". Format: Script-Literal oder Script-Variable Es ist möglich, das Datum in einem anderen Datumsformat anzugeben. Geben Sie dazu zuerst das gewünschte Datumsformat, ein Trennzeichen (: oder ;) und danach das Datum an. Die Angabe des Datumsformats ist optional.
Aktion	Kennzeichen für das Hinzufügen oder Entfernen. Format: AE-Name, Script-Literal oder Script-Variable Erlaubte Werte: "ON" (Standardwert), "OFF" "ON" - Datum/Zeitraum hinzufügen "OFF" - Datum/Zeitraum entfernen

Anmerkungen

Mit der Script-Anweisung fügen Sie einem Kalender-Objekt ein Datum/Zeitraum hinzu bzw. entfernen ein Datum/Zeitraum aus einem Kalender.

Soll nur ein einzelnes Datum hinzugefügt oder entfernt werden, entfällt die Angabe von *Datum2*. Für einen Zeitraum müssen *Datum1* und *Datum2* angegeben werden.

Die Angabe des Datumsformats ist optional. Wird kein Datumsformat verwendet, muss das Datum in den Formaten "JJMMTT" oder "JJJJMMTT" angegeben werden.

Existiert der übergebene Kalenderbegriff nicht, so legt die Script-Anweisung einen neuen Kalenderbegriff des Typs "Statisch" an. Als Gültigkeitsbereich wird standardmäßig das aktuelle Jahr minus NOW_MINUS und plus NOW_PLUS genommen. Bei beiden Einstellungen handelt es sich um Keys, die der Administrator in der Variablen UC_CLIENT_SETTINGS konfiguriert. Sie können den Gültigkeitszeitraum jedoch nachträglich mit der Script-Funktion MODIFY_OBJECT ändern.

Die Script-Anweisung ignoriert, wenn beim Hinzufügen ein Datum/Zeitraum bereits existiert bzw. wenn beim Entfernen ein Datum/Zeitraum nicht existiert.

Sie können mit der Script-Funktion VALID_CALE prüfen, ob ein Datum in einem Kalenderbegriff enthalten ist.

Beachten Sie, dass bei der Durchführung der Script-Anweisung keine Kalenderberechnung angestoßen wird! Das Kalender-Objekt muss manuell geöffnet und gespeichert werden!

Beispiele

Im ersten Beispiel wird dem Kalender "BEREITSCHAFT" und Kalenderbegriff "BU" ein Datum hinzugefügt.

```
:SET_CALE BEREITSCHAFT, "BU", "JJ-MM-TT:01-12-24"
```

Das zweite Beispiel entfernt den heutigen Tag aus diesem Kalender.

```
:SET  &HEUTE# = SYS_DATE()
:SET_CALE  "BEREITSCHAFT", "BU", &HEUTE#,, OFF
```

Das dritte Beispiel ermittelt Beginn und Ende der aktuellen Woche. Danach wird der Zeitraum in den Kalender eingetragen.

```
:SET  &HEUTE# = SYS_DATE()
:SET  &BEGINN# = FIRST_OF_PERIOD (&HEUTE#, "WW")
:SET  &ENDE# = LAST_OF_PERIOD (&HEUTE#, "WW")
:SET_CALE BEREITSCHAFT, BU, &BEGINN#, &ENDE#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
VALID_CALE	Prüft, ob ein Datum in einem Kalenderbegriff enthalten ist.

Script-Sprachmittel - Objekte lesen oder verändern
Datums- , Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.13 :SET_CONDITION

Script-Anweisung: Setzt die früheste Startzeit in Workflows.

Syntax

:SET_CONDITION ***Bedingung****[= Wert]*

Syntaxteil	Beschreibung/Format

Bedingung	Startbedingung, die gesetzt werden soll. Format: Script-Literal oder Script-Variable Erlaubte Werte: "EARLIEST_START_TIME", "JOBP_EARLIEST_START_TIME" "EARLIEST_START_TIME" = Früheste Startzeit einer Aufgabe im Workflow. "JOBP_EARLIEST_START_TIME" = Früheste Startzeit eines Workflows.
Wert	Früheste Startzeit, die gesetzt werden soll. Format: Script-Literal oder Script-Variable Format: "TT/HH:MM" Standardwert: "00/00:00"

Anmerkungen

Die Script-Anweisung funktioniert nur in den Aufgaben eines Workflows.

Die früheste Startzeit, die mit der Funktion gesetzt wird, gilt nur für die aktuelle Durchführung des Objektes. Der Zeitpunkt, der in den Eigenschaften der Aufgabe einstellbar ist, wird dadurch nicht verändert. Die früheste Startzeit kann mit der Script-Funktion GET_CONDITION auch zur Laufzeit ermittelt werden.

Für das Setzen der frühesten Startzeit gelten folgende Zusammenhänge:

- Mit "EARLIEST_START_TIME" setzt die Script-Anweisung die früheste Startzeit der Aufgabe, in deren Script sie aufgerufen wird. Die Script-Anweisung ist nicht erlaubt, wenn für die Aufgabe die Option "Zur Laufzeit generieren" (Registerkarte **Attribute**) definiert wurde. In diesem Fall wurde die Aufgabe bereits gestartet, wenn das Script prozessiert wird.
- Wird die Script-Anweisung mit "JOBP_EARLIEST_START_TIME" im Script einer Aufgabe verwendet, setzt sie die früheste Startzeit des Workflows. Auch hier ist die Script-Anweisung bei "Zur Laufzeit generieren" nicht erlaubt.
- Im Script eines Workflows kann die Script-Anweisung mit "JOBP_EARLIEST_START_TIME" nur verwendet werden, um die früheste Startzeit eines übergeordneten Workflows zu setzen. Existiert kein übergeordneter Workflow, tritt ein Fehler auf.

Bei der frühesten Startzeit des Workflows handelt es sich um jenen Zeitpunkt, der im START-Kästchen des Workflows hinterlegt wurde.

Beispiel

Im Beispiel wird die Script-Anweisung im Script einer Aufgabe verwendet, die in einem Workflow läuft. Es wird die früheste Startzeit der Aufgabe und des Workflows gesetzt und im Aktivierungsprotokoll ausgegeben.

```
:SET_CONDITION "EARLIEST_START_TIME"="00/10:19"
:SET_CONDITION "JOBP_EARLIEST_START_TIME"="00/10:18"
:SET &RETJOBS# = GET_CONDITION ("EARLIEST_START_TIME")
:SET &RETJOBP# = GET_CONDITION ("JOBP_EARLIEST_START_TIME")
:PRINT "Neue früheste Startzeit:", &RETJOBS#
:PRINT "Neue früheste Startzeit des Workflows:", &RETJOBP#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
GET_CONDITION	Ermittelt die früheste Startzeit in Workflows.

Script-Sprachmittel - Objekte lesen oder verändern
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.14 :XML_CLOSE

Script-Anweisung: Schließt ein XML-Dokument.

Syntax

:XML_CLOSE

Anmerkungen

Diese Script-Anweisung schließt ein XML-Dokument nach der Bearbeitung.

Da immer nur ein XML-Dokument zur gleichen Zeit geöffnet sein darf, ist es erst nach dem Aufruf dieser Script-Anweisung möglich, mit XML_OPEN ein anderes XML-Dokument zur Bearbeitung zu öffnen.

Diese Script-Anweisung wurde in einer früheren Version der Automation Engine von :XML_CLOSE_DOCU auf :XML_CLOSE umbenannt. Die alte Schreibweise wird jedoch weiterhin unterstützt.

Beispiel

Die Registerkarte "Details" einer strukturierten Dokumentation wird nach erfolgter Bearbeitung geschlossen.

```
:SET &XMLDOCU# = XML_OPEN("MAWI.TAG", "@Details")
!...
:XML_CLOSE
```

Siehe auch:

Script-Sprachmittel	Beschreibung
XML_OPEN	Öffnet ein XML-Dokument für die Bearbeitung.

Script-Sprachmittel - Objekte lesen oder verändern
Strukturierte Dokumentation
www.w3c.org/TR/xmlbase
Beispielsammlung
Ermittlung der Fehlermeldung und -nummer
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.15 GET_ATT

Script-Funktion: Liefert den Wert von Attributen einer Aufgabe während der Generierung.

Syntax

GET_ATT*(Attribut)*

Syntaxteil	Beschreibung/Format
Attribut	Name des Attributes, dessen Wert ausgelesen werden soll. Format: AE-name, Script-Literal oder Script-Variable

Rückgabewert
Wert des angegebenen Attributes.

Anmerkungen

Die Script-Funktion GET_ATT ermöglicht es, das angegebene *Attribut* eines Objektes während der Generierung auszulesen. Dabei können nur die Attribute verwendet werden, die zum jeweiligen Objekt gehören.

Im Benutzerhandbuch finden Sie eine Liste der Attribute aller Objekte mit ihren möglichen Werten.

Beachten Sie, dass der Wert eines Attributes auch ein Leerzeichen " " sein kann! Das tritt dann auf, wenn kein Wert gesetzt wurde (Beispiel: das Textfeld für den Archivbegriff ist leer).

Läuft die Aufgabe in einer AgentenGruppe, so gibt GET_ATT(HOST) den Namen jenes Agenten zurück, auf dem die Aufgabe tatsächlich durchgeführt wird und nicht den AgentenGruppenname.

Wenn Sie im Objekt als Attribut den Namen einer Script- bzw. Objektvariable verwenden (siehe: Attribute lesen und verändern), dann wird beim Auslesen des Attributes mit GET_ATT() nicht der Variablenname sondern der Inhalt der eingetragenen Variable zurückgeliefert. Wollen Sie nicht, dass Variablen aufgelöst werden, so ist das Sprachmittel GET_ATT_PLAIN zu verwenden.

Beispiele

Das Beispiel ermittelt den ersten Archivbegriff eines Objektes und übergibt ihn an eine Script-Variable.

```
:SET &START# = GET_ATT(ARCHIVE_KEY1)
```

Auch die Verwendung einer Script-Variablen innerhalb der Funktion ist möglich.

```
:SET &ATT# = "JOBREPORT_FILE"
:SET &START# = GET_ATT(&ATT#)
```

In diesem Beispiel wird die Funktion verwendet, um eine Bedingung zu formulieren.

```
:IF GET_ATT(GROUP) = " "
!...
:ENDIF
```

Ein Beispiel für die Verwendung einer Script-Variablen im Attribut-Feld eines Objektes. In einem FileTransfer-Objekt wird als Zieldatei die Variable "&DST#" eingetragen. Mit folgendem Script wird das Attribut gesetzt, der Variable ein Wert zugewiesen und das Attribut ausgelesen. Als Wert wird nicht der Variablenname "&DST#" zurückgeliefert, sondern der Wert der Variable "C:\Temp\test2.txt".

```
:PUT_ATT FT_DST_FILE = "&&DST#"
 :SET &DST# = "C:\Temp\test2.txt"
 :SET &ZIEL# = GET_ATT(FT_DST_FILE)
 :PRINT "Zieldatei: &ZIEL#"
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:ADD_ATT	Fügt einem Benachrichtigungs-Objekt Empfänger zur Laufzeit hinzu.
:REMOVE_ATT	Entfernt in einem Benachrichtigungs-Objekt Empfänger zur Laufzeit.
:PUT_ATT	Ändert den Wert eines Attributes während der Generierung.
:PUT_ATT_APPEND	Erweitert den Meldungstext einer Benachrichtigung zur Laufzeit.
GET_ATT_PLAIN	Liefert den Wert von Attributen einer Aufgabe während der Generierung, wobei Variablen nicht aufgelöst werden.
GET_ATT_SUBSTR	Liefert einen Teil des Meldungstextes einer Benachrichtigung.

Script-Sprachmittel - Objekte lesen oder verändern
Beispielsammlung
Benachrichtigung mit variablem Meldungstext
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.16 GET_ATT_PLAIN

Script-Funktion: Liefert den Wert von Attributen einer Aufgabe während der Generierung, wobei Variablen nicht aufgelöst werden.

Syntax

GET_ATT_PLAIN*(Attribut)*

Syntaxteil	Beschreibung/Format
Attribut	Name des Attributes, dessen Wert ausgelesen werden soll. Format: AE-name, Script-Literal oder Script-Variable

Rückgabewert
Wert des angegebenen Attributes.

Anmerkungen

Mit dieser Script-Funktion ist es möglich Attribute des Objektes während der Generierung auszulesen. Das Verhalten ähnelt dem Sprachmittel GET_ATT. Unterschied ist jedoch, dass Script-, Objekt- und vordefinierte Variablen im Attribut unersetzt zurückgeliefert werden. Es wird daher der Name der Variablen ermittelt und nicht deren Wert. Bei GET_ATT werden diese Variablen immer aufgelöst.

Platzhalter für Variablen-Objekte die in { } - Klammern stehen, werden bei keinen der zwei Sprachmittel ersetzt zurückgeliefert.

Ab Version 9.00A liest der Attribut-Dialog die Attribute mit dieser Script-Funktion aus. Im Dialog wird nun immer der tatsächliche Attribut-Inhalt angezeigt und nicht mehr der ersetzte Wert von Variablen.

Beachten Sie auch die Hinweise die in der Beschreibung von GET_ATT zu finden sind.

Beispiel

In folgendem Beispiel wird der Name einer Script-Variable in ein Attribut eingetragen und dieser ein Wert zugewiesen. Anschließend wird der Attribut-Inahlt mit und ohne ersetzten Variablenwert ermittelt und die Information im Aktivierungsprotokoll ausgegeben.

```
:PUT_ATT INT_ACCOUNT = "&&test#"
:SET &test# = "test"
:SET &att# = GET_ATT(INT_ACCOUNT)
:SET &attplain# = GET_ATT_PLAIN(INT_ACCOUNT)
:PRINT "Variablenname = &attplain#"
:PRINT "Variablenwert = &att#"
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:ADD_ATT	Fügt einem Benachrichtigungs-Objekt Empfänger zur Laufzeit hinzu.
:REMOVE_ATT	Entfernt in einem Benachrichtigungs-Objekt Empfänger zur Laufzeit.
:PUT_ATT	Ändert den Wert eines Attributes während der Generierung.
:PUT_ATT_APPEND	Erweitert den Meldungstext einer Benachrichtigung zur Laufzeit.
GET_ATT	Liefert den Wert von Attributen einer Aufgabe während der Generierung
GET_ATT_SUBSTR	Liefert einen Teil des Meldungstextes einer Benachrichtigung.

Script-Sprachmittel - Objekte lesen oder verändern
Beispielsammlung
Benachrichtigung mit variablem Meldungstext
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.17 GET_ATT_SUBSTR

Script-Funktion: Liefert einen Teil des Meldungstextes einer Benachrichtigung.

Syntax

GET_ATT_SUBSTR*(CALL_TEXT, Beginn, Länge)*

Syntaxteil	Beschreibung/Format
CALL_TEXT	Name des Attributes, von dessen Wert ein Teil gelesen werden soll. Format: AE-name und Script-Variable Erlaubter Wert: "CALL_TEXT"
Beginn	Stelle, ab der gelesen werden soll. Format: Zahl oder Script-Variable
Länge	Anzahl der Zeichen, die gelesen werden sollen. Format: Zahl oder Script-Variable

Rückgabewerte
Teil des Meldungstextes. " " - Der angegebene Bereich liegt nicht innerhalb des Meldungstextes.

Anmerkungen

Diese Script-Funktion kann nur in der Script-Registerkarte eines Benachrichtigungs-Objektes eingesetzt werden. Sie ermittelt einen Teil des Meldungstextes einer Benachrichtigung.

Wenn vor der Script-Funktion die Anweisung :PUT_ATT_APPEND steht, wird der Teil aus dem erweiterten Meldungstext entnommen.

Beispiel

Im Beispiel werden die ersten 20 Zeichen des Meldungstextes gelesen.

```
:SET &AUSSCHNITT# = GET_ATT_SUBSTR(CALL_TEXT,1,20)
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:ADD_ATT	Fügt einem Benachrichtigungs-Objekt Empfänger zur Laufzeit hinzu.
:REMOVE_ATT	Entfernt in einem Benachrichtigungs-Objekt Empfänger zur Laufzeit.
:PUT_ATT	Ändert den Wert eines Attributes während der Generierung.
:PUT_ATT_APPEND	Erweitert den Meldungstext einer Benachrichtigung zur Laufzeit.
GET_ATT	Liefert den Wert von Attributen einer Aufgabe während der Generierung.

Script-Sprachmittel - Objekte lesen oder verändern

Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.18 GET_CONDITION

Script-Funktion: Ermittelt die früheste Startzeit in Workflows.

Syntax

GET_CONDITION*(Bedingung)*

Syntaxteil	Beschreibung/Format
Bedingung	Startbedingung, die ermittelt werden soll. Format: Script-Literal oder Script-Variable Erlaubte Werte: "EARLIEST_START_TIME", "JOBP_EARLIEST_START_TIME" "EARLIEST_START_TIME" = Früheste Startzeit einer Aufgabe im Workflow. "JOBP_EARLIEST_START_TIME" = Früheste Startzeit eines Workflows.

Rückgabewert
Früheste Startzeit im Format "TT/HH:MM" der Aufgabe bzw. des Workflows.

Anmerkungen

Die Script-Funktion kann nur im Rahmen eines Workflows eingesetzt werden.

Die früheste Startzeit ist in den Eigenschaften der Aufgaben und im START-Kästchen eines Workflows festlegbar. Sie kann mit der Script-Anweisung :SET_CONDITION auch zur Laufzeit gesetzt werden.

Für das Ermitteln der frühesten Startzeit gelten folgende Zusammenhänge:

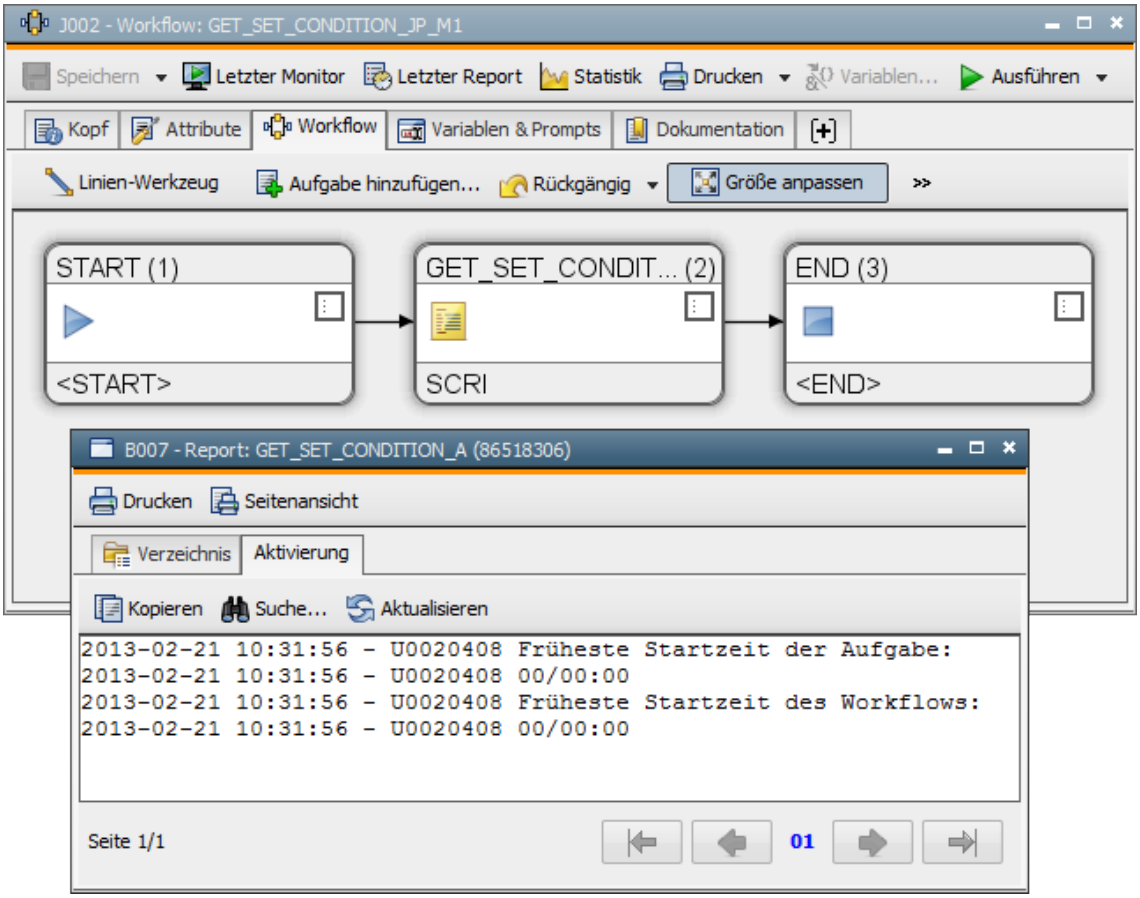
- Mit "EARLIEST_START_TIME" ermittelt die Script-Funktion die früheste Startzeit der Aufgabe, in deren Script sie aufgerufen wird.
- Wird die Script-Funktion mit "JOBP_EARLIEST_START_TIME" im Script einer Aufgabe verwendet, gibt sie die früheste Startzeit des Workflows zurück.
- Im Script eines Workflows kann die Script-Funktion mit "JOBP_EARLIEST_START_TIME" nur verwendet werden, wenn dieser Workflow in einem übergeordneten Workflow läuft. Zurückgeben wird die früheste Startzeit des übergeordneten Workflows. Existiert kein übergeordneter Workflow, tritt ein Fehler auf.

Bei der frühesten Startzeit des Workflows handelt es sich um jenen Zeitpunkt, der im START-Kästchen des Workflows hinterlegt wurde.

Beispiel

Im Beispiel wird die Script-Funktion im Script einer Aufgabe verwendet, die in einem Workflow läuft. Es wird die früheste Startzeit der Aufgabe und des Workflows ermittelt und im Aktivierungsprotokoll ausgegeben.

```
:SET   &RETJOBS# = GET_CONDITION ("EARLIEST_START_TIME")
:SET   &RETJOBP# = GET_CONDITION ("JOBP_EARLIEST_START_TIME")
:PRINT  "Früheste Startzeit der Aufgabe:", &RETJOBS#
:PRINT  "Früheste Startzeit des Workflows:", &RETJOBP#
```



Siehe auch:

Script-Sprachmittel	Beschreibung
:SET_CONDITION	Setzt die früheste Startzeit in Workflows.

Script-Sprachmittel - Objekte lesen oder verändern
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.19 GET_CONNECTION

Script-Funktion: Liest Informationen aus einem DB Verbindungs-Objekt.

Syntax

GET_CONNECTION*(Objektname, Attribut)*

Syntaxteil	Beschreibung/Format
Objektname	Name eines Verbindungs-Objektes vom Typ DB Format: AE-Name, Script-Literal oder Script-Variable
Attribut	Wert, der aus dem Objekt gelesen werden soll Format: AE-Name, Script-Literal oder Script-Variable Erlaubte Werte: • **db.type** = Datenbank-Typ Mögliche zurückgelieferte Werte: ◦ "MSSQL" - MS SQL Server ◦ "DB2" - IBM DB2 ◦ "ORACLE" - Oracle ◦ "SYBASE" - SyBase ◦ "MYSQL" - MyQL ◦ "INFORMIX" - Informix ◦ "ORACLE_OCI" - Oracle OCI • **db.server** = Name des DB-Servers • **db.port** = Portnummer des DB-Servers • **db.name** = Name der Datenbank • **DbUserid** = Datenbank-Benutzer • **DbPassword** = Password des DB-Benutzers • **Login** = Name des Alternativ-Login-Objektes ⚠ Bitte beachten Sie: Bei Datenbanken, in denen die Groß- und Kleinschreibung eine Rolle spielt, müssen Sie den Wert **DbUserid** in der hier gezeigten Schreibweise verwenden.

Rückgabewerte
Gewünschter Wert des CONN-Objektes

Anmerkungen

Die Script-Funktion liest einen bestimmten Wert aus einem Verbindungs-Objekt vom Typ DB und liefert diesen als Rückgabewert zurück.

Wurde das angegebene Objekt nicht gefunden, entspricht das Verbindungs-Objekt nicht dem Typ DB oder ist das spezifizierte Attribut ungültig, so kommt es zu einem Laufzeitfehler.

Beispiel

Folgendes Beispiel ermiitelt den Datenbank-Typ des Verbindungs-Objektes CONN.DB und schreibt diesen in den Aktivierungsreport.

```
:SET &VAR# = GET_CONNECTION(CONN.DB, db.type)
:P&VAR#
```

Im Aktivierungsprotokoll ist anschließend folgende Zeile zu finden:

```
2013-01-30 12:45:21 - U0020408 MSSQL
```

Siehe auch:

Script-Sprachmittel	Beschreibung
GET_ATT	Liefert den Wert von Attributen einer Aufgabe während der Generierung.
GET_LOGIN	Liest Informationen aus Login-Objekten.

Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.20 GET_LOGIN

Script-Funktion: Liest Informationen aus Login-Objekten.

Syntax

GET_LOGIN*(Login-Objekt, Name, Typ, Information)*

Syntaxteil	Beschreibung/Format
Login-Objekt	Name eines Login-Objekts Format: Script-Variable, AE-Name oder Script-Literal
Name	Name des Eintrages im Login-Objekt Format: Script-Variable, AE-Name oder Script-Literal
Typ	Typ des Login-Eintrages Format: Script-Variable, AE-Name oder Script-Literal
Information	Auszulesende Information Format: AE-Name Erlaubte Werte: LOGIN_INFO - Login-Info (Benutzer, Domäne) auslesen PASSWORD - Passwort

Rückgabewert

Entsprechende Information

20476 - Das angegebene Objekt existiert nicht oder entspricht nicht dem Typ LOGIN.
20482 - Angegebener Name im Login-Objekt nicht gefunden.
20493 - Ungültige Angabe der auszulesenden Information.

Anmerkungen

Die Script-Funktion liefert die Login-Info oder das Passwort eines bestimmten Eintrags im angegebenen *Login-Objekt*. Welcher dabei Eintrag verwendet wird, bestimmen die Parameter *Name* und *Typ*.

⚠ Dabei können nur Login-Einträge von Backend-Systemen ausgelesen werden! Achten Sie darauf, dass der angegebene *Typ* auch in der Variable UC_LOGIN_TYPES vorkommt.

Das Passwort wird verschlüsselt zurückgeliefert. Sie können das Passwort im Jobmelder-Aufruf (Start-Parameter CMD) verwenden, wo es auch entschlüsselt wird .

Bei *Name* kann auch "*" angegeben werden, wobei dieser unter einfache oder doppelte Hochkommata zu setzen ist. Dabei handelt es sich nicht um einen Filter, sondern dem Eintrag "*" im Login-Objekt.

Beispiel

In folgendem Beispiel wird die Login-Info und das Passwort des Backendsystem-Eintrages SVN aus dem Login-Objekt gelesen. Anschließend wird ein Kommando mit Angabe der Login-Daten über den Jobmelder durchgeführt.

```
:SET &LOGIN# = GET_ATT(LOGIN)
:SET &LI# = GET_LOGIN(&LOGIN#,SVN,SVN,LOGIN_INFO)
:SET &PW# = GET_LOGIN(&LOGIN#,SVN,SVN,PASSWORD)

&UC_JOBMD CMD="&SVN_CMD# checkout ""&SVN_URI#"" ""&SVN_DIR#"" --
username &LI# --password &PW#"
```

Siehe auch:

Script-Sprachmittel - Datensequenzen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.21 GET_OBJECT_TYPE

Script-Funktion: Liefert den Objekttyp der Aufgabe.

Syntax

GET_OBJECT_TYPE*(Objektname)*

Syntaxteil	Beschreibung/Format
Objektname	Name des Objektes dessen Typ ermittelt werden soll Format: Script-Literal oder Script-Variable

Rückgabewerte
Kurzform des Objekttyps "20223" - Das Objekt existiert nicht.

Anmerkungen

Die Script-Funktion sucht das Objekt zuerst in dem Mandanten in dem das Script ausgeführt wird und danach im Systemmandanten 0000.

Mit der Script-Anweisung :ON_ERROR legen Sie die Reaktion auf einen fehlerhaften Objektnamen fest. Sie können den Fehler mit den Script-Funktionen zur Fehlerbehandlung analysieren. Das Script wird weiter prozessiert. Es besteht aber auch die Möglichkeit, die Prozessierung des Scripts abzubrechen.

Beispiel

Das Beispiel liefert "JOBP", da es sich bei MAWI.TAG um einen Workflow handelt.

```
:SET &OBJEKT_TYP# = GET_OBJECT_TYPE("MAWI.TAG")
```

Siehe auch:

Script-Sprachmittel - Objekte lesen oder verändern
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.22 GET_OH_IDNR

Script-Funktion: Liefert die interne Nummer eines Objektes.

Syntax

GET_OH_IDNR*(Objektname, Mandant)*

Syntaxteil	Beschreibung/Format
Objektname	Name des Objektes Format: Script-Literal oder Script-Variable
Mandant	Nummer des Mandanten in dem sich das Objekt befindet Format: Zahl oder Script-Variable

Rückgabewert
Interne Nummer des Objektes "0" - Das Objekt existiert nicht.

Anmerkungen

Jedes Objekt erhält beim Anlegen eine interne Nummer.

Beispiel

Das folgende Beispiel liefert die interne Nummer des Objektes "MAWI.TAG", das sich im Mandanten 98 befindet.

```
:SET &RET# = GET_OH_IDNR("MAWI.TAG",98)
```

Siehe auch:

Script-Sprachmittel - Objekte lesen oder verändern
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.23 GET_PUBLISHED_VALUE

Script-Funktion: Wert einer Objekt- bzw. PromptSet-Variablen einer bestimmten Aufgabe ermitteln.

Syntax

GET_PUBLISHED_VALUE*(RunID, Variablenname)*

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) der Aufgabe Format: Script-Literal oder Script-Variable
Variablenname	Name einer Objekt- bzw. PromptSet-Variable der angegebenen Aufgabe (ohne führendes &). Format: AE-Name Bei Arrays sind am Ende des Namens zusätzlich die leeren Indexklammern [ ] anzugeben.

Rückgabewert
Wert der Objekt- bzw. PromptSet-Variable

Anmerkungen

Diese Script-Funktion funktioniert sowohl in Zusammenhang mit :SET als auch :FILL. :SET ist zu verwenden, wenn der Wert einer normalen Objekt-/PromptSet-Variable oder eines einzelnen PromptSet-Array Elementes ausgelesen wird. :FILL dient dazu ein komplettes PromptSet-Array auszulesen.

Die Möglichkeit, PromptSet-Variablen als Arrays zu definieren, besteht nur bei den Checklist- und Checkbox-Elementen.

Beispiele

In folgendem Beispiel wird ein Objekt aktiviert und anschließend dessen Objektvariable &VARIABLE1# ausgelesen.

```
:SET &RUNID# = ACTIVATE_UC_OBJECT(&OBJ#,WAIT)
:SET &VAR# = GET_PUBLISHED_VALUE(&RUNID#,VARIABLE1#)
:PRINT "&&VARIABLE1# = &VAR#"
```

Im zweiten Beispiel lesen wir die PromptSet-Variable &CHECKLIST1# aus, die als Array definiert wurde. Dabei ermitteln wir im ersten Schritt nur den Wert des ersten Array-Elementes.

```
:SET &RUNID# = ACTIVATE_UC_OBJECT(&OBJ#,WAIT)
:SET &VAR# = GET_PUBLISHED_VALUE(&RUNID#,CHECKLIST1#[1])
:PRINT "&&CHECKLIST1#[1] = &VAR#"
```

Folgendes Szenario zeigt die Ermittlung eines kompletten PromptSet-Arrays der aktivierten Aufgabe. Anschließend werden die einzelnen PromptSet-Elemente im Aktivierungsreport ausgegeben.
Würde in diesem Fall das PromptSet-Array die Größe des Script-Arrays übersteigen, so werden nur jene Elemente gespeichert, die Platz finden.

```
:DEFINE &ARRAY#, string, 10
:SET &RUNID# = ACTIVATE_UC_OBJECT(&OBJ#,WAIT)
:FILL &ARRAY# = GET_PUBLISHED_VALUE(&RUNID#,CHECKLIST1#[])
:SET &LEN# = LENGTH(&ARRAY#[])
:SET &VAR# = 1

:WHILE &VAR# LE &LEN#
 :PRINT "&&ARRAY#[&VAR#] = &ARRAY#[&VAR#]"
 :SET &VAR# = &VAR# + 1
:ENDWHILE
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:FILL	Speichert mehrere Werte in ein Script-Array.
:PUBLISH	Script-Variablen und Arrays als Objektvariablen definieren.

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.24 GET_STATISTIC_DETAIL

Script-Funktion: Ermittelt Details aus einem Statistiksatz eines aktivierbaren Objektes.

Syntax

GET_STATISTIC_DETAIL(*[RunID]* **, *Detail*** *[, Objektname]***)**

Syntaxteil	Beschreibung/Format
RunID	10-stellige laufende Nummer (RunID) der Durchführung Format: Script-Variable oder Zahl
Detail	Information, die aus dem Statistiksatz ermittelt werden soll. Format: AE-Name, Script-Literal oder Script-Variable Bei den Anmerkungen finden Sie eine Tabelle mit den erlaubten Werten.
Objektname	Name des Objektes dessen Statistiksatz ausgelesen werden soll. Format: Script-Literal oder Script-Variable

Rückgabewert
Detail aus einem Statistiksatz

Anmerkungen

⚠ Beachten Sie, dass Benutzer die Berechtigung "S" für das jeweilige Objekt benötigen, um diese Script-Funktion ausführen zu können.

Die Script-Funktion kann in den folgenden Parameterkonstellationen genutzt werden:

- **RunID, Detail** bzw. **RunID, Detail, Objektname**
  Über die RunID wird der auszulesende Statistiksatz eindeutig identifiziert.
- **Detail, Objektname**
  Es wird der letzte Statistiksatz des angegebenen Objektes ausgelesen.
- **Detail**
  Der aktuelle Statistiksatz des Objektes aus dem die Script-Funktion aufgerufen wird, dient zur Detailermittlung. Dies bedeutet auch, dass bestimmte Werte wie z.B. die Laufzeit bei der Scriptprozessierung noch nicht verfügbar sind.

⚠ Gehen Sie behutsam bei der Formulierung der Script-Funktion um! Für Werte, die zum Zeitpunkt der Scriptausführung noch nicht ermittelbar sind, werden folgende Standardwerte zurückgegeben:

- Für Zeichenfolgen: " "
- Für Zahlen: 0
- Für Datums- und Zeitstempel: 0000-00-00 00:00:00

Bei den Werten kann es sich aber auch um Werte von ausgelesenen Statistikdetails handeln (z.B: Rückgabewert 0)!

⚠ Wird kein Statistiksatz gefunden, so liefert die Script-Funktion einen Leerstring zurück. Es kommt nicht zum Abbruch des Scripts! Der Fehler kann jedoch mit dem Sprachmittel :ON_ERROR abgefangen werden.

Der zweite Parameter erfordert die Angabe des Statistikdetails, das Sie auslesen möchten. Die folgende Tabelle zeigt Ihnen welche Werte ermittelt werden können:

Detail	Beschreibung
ACCOUNT	Kostenstelle
ACTIVATION_TIME	Zeitpunkt der Aktivierung im Format "JJJJ-MM-TT HH:MM:SS"
ARCHIVE_KEY_1	Archivbegriff 1
ARCHIVE_KEY_2	Archivbegriff 2
CANCEL_FLAG	Abbruch der Aufgabe Rückgabewerte: " " - Aufgabe wurde nicht abgebrochen "m" - Aufgabe wurde durch einen manuellen Eingriff abgebrochen
CHECK_COUNT	Anzahl der Überprüfungen des Ereignis-Objektes
COMPRESSION_RATE	Komprimierungslevel Rückgabewerte: "0" - Keine "1" - Normal "2" - Stark " " - Standardwert
CPU_TIME	Benutzte CPU-Zeit
DST_CODE_TABLE	Name der Ziel-CodeTabelle bei FileTransfers
DST_FILE_ ATTRIBUTES	Dateiattribute für die Zieldatei bei FileTransfers
DST_FILE_NAME	Name der Zieldatei bei FileTransfers
DST_HOST	Name des Zielagenten bei FileTransfers und Jobs
DST_HOST_TYPE	Hosttyp des Zielagenten bei FileTransfers Rückgabewerte: "BS2000", "GCOS8", "MPE", "MVS", "NSK", "OS400", "UNIX", "VMS" und "WINDOWS"
DST_LOGIN_INFO	Gesamte Login Info aus dem Ziel-Login-Objekt des FileTransfers
DST_LOGIN_NAME	Name des Ziel-Login-Objektes
DURATION	Laufzeit in Sekunden
END_TIME	Endezeit des Objektes im Format "JJJJ-MM-TT HH:MM:SS"
EVENTID	Erste RunID von Dateisystem- und Konsol-Ereignissen
FILE_SIZE	Anzahl Bytes der übertragenen Datei
IO_COUNT	Anzahl der I/Os
KERNEL_TIME	Benutzte Kernel-Zeit
LAST_ERR_INS	Meldungsinsert
LAST_ERR_NR	Nummer des zuletzt aufgetretenen Fehlers
LAST_RESTART_ POINT	Zuletzt durchlaufener Wiederanlaufpunkt
LDATE	Logisches Datum im Format "JJJJ-MM-TT HH:MM:SS"

MOD_COUNT	Anzahl der Änderungen am Objekt
NAME	Name des Objektes
OBJECT_TYPE	Objekttyp
OCCURENCE_COUNT	Anzahl der eingetretenen Ereignisse (bei Ereignis-Objekten)
PARENT_ACT	RunID der übergeordneten Aufgabe (Activator)
PARENT_PRC	RunID der übergeordneten Aufgabe (Processor)
POSTSCRIPT_START_TIME	Startzeit des Post-Scriptes im Format "JJJJ-MM-TT HH:MM:SS"
PROCESS_ID	TSN / Prozess-ID
RECORDS	Text-FileTransfers: Anzahl der übertragenen Zeilen / Records. Binär-FileTransfers: 0
REFERENCE_NR	Referenz-RunID beim Wiederanlauf
RESTART	Wiederanlauf Rückgabewerte: "Y" - Es handelt sich um einen Wiederanlauf. "N" - Die Durchführung ist kein Wiederanlauf.
RESTART_POINT	Wiederanlaufpunkt von dem die Aufgabe gestartet ist
RETURN_CODE	Rückgabewert
RUNID	RunID des gewählten Statistiksatzes
SRC_CODE_TABLE	Name der Quell-CodeTabelle bei FileTransfers
SRC_FILE_ATTRIBUTES	Dateiattribute für die Quelldatei bei FileTransfers
SRC_FILE_NAME	Name der Quelldatei bei FileTransfers
SRC_HOST	Name des Quellagenten bei FileTransfers
SRC_HOST_TYPE	Hosttyp des Quellagenten bei FileTransfers Rückgabewerte: "BS2000", "GCOS8", "MPE", "MVS", "NSK", "OS400", "UNIX", "VMS" und "WINDOWS"
SRC_LOGIN_INFO	Gesamte Login Info aus dem Quell-Login-Objekt des FileTransfers
SRC_LOGIN_NAME	Name des Quell-Login-Objektes
START_TIME	Startzeit des Objektes im Format "JJJJ-MM-TT HH:MM:SS"
STATUS	Status (System-Returncode) der Durchführung (z.B. "1850")
TRANSFERRED_BYTE_COUNT	Anzahl der übertragenen Bytes
USER_ID	Name des Benutzers im Format "NAME/ABTEILUNG"
USER_TIME	Benutzte User-Zeit

Beispiel

Die folgende Zeile wird in einem FileTransfer-Objekt verwendet. Sie liest den Namen der zu übertragenden Datei aus der aktuellen Durchführung aus.

```
:SET &QUELLDATEI# = GET_STATISTIC_DETAIL(,SRC_FILE_NAME)
```

Im zweiten Beispiel wird die Startzeit der übergeordneten Aufgabe ausgelesen.

```
:SET &NAME# = SYS_ACT_PARENT_NAME()
:SET &START# = GET_STATISTIC_DETAIL(,START_TIME, &NAME#)
```

Hier ermittelt die Script-Funktion den Aktivierungszeitpunkt des Objektes MAWI.TAG.

```
:SET&RUNNR# = GET_UC_OBJECT_NR("MAWI.TAG")
:SET&ACTIVATION# = GET_STATISTIC_DETAIL(&RUNNR#, ACTIVATION_TIME)
```

Siehe auch:

Script-Sprachmittel - Objekte lesen oder verändern
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.25 GET_SYNC

Script-Funktion: Fragt den aktuellen Zustand oder Wert eines Sync-Objektes ab.

Syntax

GET_SYNC*(Sync, Typ)*

Syntaxteil	Beschreibung/Format
Sync	Name eines Sync-Objektes, dessen aktueller Zustand oder Wert ermittelt werden soll. Format: AE-name, Script-Literal oder Script-Variable
Typ	Angabe, ob der aktuelle Zustand oder der aktuelle Wert ermittelt werden soll. Format: AE-Name, Script-Literal oder Script-Variable Erlaubte Werte: "STATE" und "VALUE" "STATE" - Aktueller Zustand "VALUE" - Aktueller Wert

Rückgabewerte
Aktueller Zustand des Sync-Objektes. Aktueller Wert des Sync-Objektes.

Anmerkungen

Die Script-Funktion liefert die aktuellen Einstellungen eines Sync-Objektes, die auch in der Registerkarte "Attribute" zu finden sind.

Beispiel

Die folgenden Scriptzeilen nehmen Bezug auf das Beispiel Sync für Zugriffe von Jobs nutzen.

```
:SET  &ADMIN# = GET_SYNC("DB.STATUS","STATE")
:IF  &ADMIN# = "EXKLUSIV"
:  PRINT  "Der Administrator-Job belegt momentan exklusiv die
Datenbank."
:ENDIF


:SET  &RET# = GET_SYNC("DB.STATUS","VALUE")
:IF  &RET# = 0
:   PRINT  "Kein Job greift momentan auf die Datenbank zu."
:ELSE
:   PRINT  "Es greifen gerade &RET# Jobs auf die Datenbank zu."
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SET_SYNC	Führt die definierte Aktion eines Sync-Objektes aus.
:ATTACH_SYNC	Weist einer Aufgabe ein Sync-Objekt zu.

Script-Sprachmittel - Objekte lesen oder verändern
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.26 GET_VAR

Script-Funktion: Liefert den Wert eines Variablen-Objektes.

Syntax

GET_VAR*(Variable *[, [Key] [, Spalte]]**)***

Syntaxteil	Beschreibung/Format
Variable	Name eines Variablen-Objektes, deren Werte ausgelesen werden soll. Format: AE-name, Script-Literal oder Script-Variable
Key	Zeile der Variable Wert der Key-Spalte (statische Variablen) bzw. der ersten Rohspalte (dynamische Variablen) Format: Script-Literal oder Script-Variable

Spalte	Nummer der Spalte, dessen Wert(e) verwendet werden soll(en). Format: Script-Literal, Script-Variable oder Zahl ohne Hochkommata Erlaubte Werte: Statische Variablen: "KEY" (Key-Spalte), "1" bis "5" (Wertespalte 1 bis 5) Dynamische Variablen: "RESULT" (Result-Spalte), "1" bis n (Wertespalte 1 bis n)

Rückgabewerte
Wert/e der Variable " " - Der Eintrag existiert nicht bzw. enthält keinen Wert.

Anmerkungen

Mit dem Scriptsprachmittel können Werte sowohl aus statischen als auch dynamischen Variablen-Objekten gelesen werden. Beim Zugriff auf dynamische Variablen erfolgt eine Auflösung, daher wird der Wert direkt aus der Datenquelle (Datenbank, Variable, Verzeichnis) ermittelt.

⚠ Kommt es aus irgendeinem Grund zu einem Fehler bei der Auflösung von dynamischen Variablen-Objekten, so führt dies zu einem Laufzeitfehler!

Wird der Parameter *Key* nicht angegeben, so wird automatisch die erste Zeile verwendet. Bei statischen Variablen-Objekten, die als Gültigkeitsbereich "Frei gewählt" eingestellt haben, ist der Key allerdings immer anzugeben!
Fehlt die*Spalte*, so wird automatisch die Spalte **Wert 1** (bei statischen Variablen-Objekten), die Result-Spalte (Variablen-Quelle: SQL, SQL-intern, Multi) oder die Spalte **Dateiliste** (Variablen-Quelle: Dateiliste) verwendet.

Bei Variablen mit der Quelle **Dateiliste** ist Spalte entweder auszulassen oder der Wert "1" zu verwenden. Da es nur eine Spalte gibt, entspricht der angegebene Key dem zurückgelieferten Wert.

Als Wert für *Spalte* ist der Bereich von 1 (erste Wertespalte) bis zur letzten Spaltennummer der Variable erlaubt. Beachten Sie, dass statische Variablen maximal 5 Werte-Spalten besitzen. Um die Result-Spalte (dynamische Variablen außer "Dateiliste") auszulesen, ist der Wert "RESULT" für *Spalte* zu verwenden. Wird das Sprachmittel im Zusammenhang mit :FILL verwendet und keine Spalte angegeben, so werden alle Werte der Variablen-Zeile in das Script-Array gespeichert.

Das Format des Wertes entspricht dem Ausgabeformat, das über die Registerkarte **Attribute** festgelegt wurde.

Mit der Script-Funktion können auch die sogenannten Agent-Variablen ausgelesen werden.

⚠ Um auf einen Key zuzugreifen, der mit dem Zeichen & beginnt, ist dieses Zeichen doppelt anzugeben. Andernfalls wird dieser Ausdruck als Script-Variable interpretiert und versucht diese aufzulösen.

Beispiel: Zugriff auf den Key "&key" im Variablen-Objekt VARA.TEST

```
:SET &TEST# = GET_VAR(VARA.TEST,"&&key")
```

Ein Leerzeichen wird geliefert, wenn Sie mit GET_VAR auf einen Key zugreifen möchten, der nicht existiert.

⚠ Bei dynamischen Variablen ist der Wert der Result-Spalte (Parameter *Key*) in dem Format anzugeben, welches der Wert in der Datenquelle (SQL - Datenbank, MULTI - Variablen-Objekt) aufweist. Das Ausgabeformat der Variable spielt dabei keine Rolle!

Bei statischen Variablen kann für *Spalte* auch der spezielle Wert "KEY" angegeben werden. Mit diesem haben Sie die Möglichkeit zu prüfen, ob ein bestimmter Key in dem Variablen-Objekt existiert. Liefert das Sprachmittel einen leeren Wert, existiert der Key nicht. Wird der Key zurückgegeben, so ist der Eintrag vorhanden.
In folgendem Beispiel liefert das Sprachmittel "Test", wenn der Key existiert, " " wenn nicht:

```
:SET &KEY# = GET_VAR(VARA.TEST, "Test","KEY")
```

Beispiele

Das Beispiel ermittelt den Wert aus einer Variablen und übergibt ihn an eine Script-Variable.

```
:SET &TEST# = GET_VAR(MAWI_ABSCHLUSS, "Buchungstag")
```

Auch die Verwendung von Script-Variablen innerhalb der Funktion ist möglich.

```
:SET &VAR# = "MAWI_ABSCHLUSS"
:SET &GUELT# = "Buchungstag"
:SET &TEST# = GET_VAR(&VAR#, &GUELT#)
```

In diesem Beispiel wird die Funktion verwendet, um eine Bedingung zu formulieren.

```
:IF GET_VAR(MAWI_ABSCHLUSS, "BUCHUNGSTAG") = SYS_DATE("DDMMYY")
!...
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:DELETE_VAR	Löscht einen oder alle Werte aus einem statischen Variablen-Objekt.
:PUT_VAR	Speichert Werte in ein statisches Variablen-Objekt.
:SET_SCRIPT_VAR	Setzt die Werte von Script-Variablen per indirektem Zugriff.
GET_SCRIPT_VAR	Liefert die Werte von Script-Variablen per indirektem Zugriff.
PREP_PROCESS_VAR	Bereitet die Bearbeitung einer Datensequenz (Werte eines Variablen-Objektes) vor.

Script-Sprachmittel - Objekte lesen oder verändern
Agent-Variablen
Beispielsammlung:
Datenbankwartung mit Optionen
Aufruf eines MBeans
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.27 MODIFY_TASK

Script-Funktion: Ändert laufende Workflows.

Allgemeine Informationen

Die Änderung eines laufenden Workflows erfolgt in mehreren Schritten. Alle werden mit Hilfe von MODIFY_TASK durchgeführt:

1. Stoppen Sie den Workflow mit dem Status "STOP_MODIFY".
2. Nehmen Sie eine oder mehrere Änderungen vor.
3. Aktivieren Sie die Änderungen mit dem Status "COMMIT".
4. Starten Sie den Workflow mit dem Status "GO".

Die Parameter der Script-Funktion sind in einer bestimmten Reihenfolge anzuordnen. Wenn Sie daher Parameter auslassen, sind trotzdem deren Kommata zu setzen (siehe nachfolgende Beispiele).

Verwenden Sie "START" bzw. "END", um die Eigenschaften des START- bzw. Ende-Kästchens zu ändern.

Beachten Sie, dass die Registerkarte **Ergebnis** (Workflow-Eigenschaften) ab Version 9.00A nicht mehr existiert und daher mit dem Sprachmittel nicht mehr geändert werden kann!

Mit dem Sprachmittel ist es nicht möglich, die Ausführung des eigenen Workflows zu stoppen bzw. fortzusetzen und diesen in weiterer Folge zu ändern.

Rückgabewerte

Allgemein:

"0" - Die Änderung wurde erfolgreich vorgenommen.
"9" - Der Benutzer verfügt nicht über die Berechtigungen, die angegebene Aufgabe zur Laufzeit zu modifizieren.
"20591" - Der Workflow mit der RunID existiert nicht oder ist nicht mehr aktiv.
"20738" - Die Aufgabe befindet sich in einem Status in dem die Änderung nicht vorgenommen werden kann.
"20739" - Die Aufgabe kann nicht geändert werden, da sie zur Zeit von einem anderen Benutzer modifiziert wird.
"20740" - Angabe eines ungültigen Objekttyps.
"20741" - Es existiert keine Aufgabe mit dem angegebenen Namen und/oder der angegebenen laufenden Nummer.
"20742" - Der angegebene Aufgabenname passt nicht zur angegebenen laufenden Nummer.
"20743" - Die Aufgabe kann nicht geändert werden, da ihr Workflow nicht zum Editieren gestoppt wurde.
"20744" - Die Aufgabe kann nicht geändert werden, da sie sich in keinem modifizierbaren Status befindet.
"20745" - Die Änderung der Aufgabe ist nicht zulässig, da der Wert nicht erlaubt ist.
"20746" - Hinzufügen einer Abhängigkeit ist nicht möglich, da diese bereits existiert.
"20747" - Die Änderung einer Abhängigkeit ist nicht möglich, da sie sich in keinem modifizierbaren Status befindet.
"20748" - Das Hinzufügen einer Aufgabe ist nicht möglich, da sich an der Stelle bereits eine Aufgabe befindet.
"20749" - Der Objekttyp der angegebenen Aufgabe ist nicht zulässig.
"20750" - Die Funktion kann für die Aufgabe mit der RunID auf Grund eines vorigen Fehlers nicht durchgeführt werden.
"20752" - Das Ersetzen einer externen Aufgabe ist nur durch eine externe Aufgabe möglich.
"20753" - Die Ersetzung von <START> und <END> ist nicht möglich.
"20754" - <START> darf keinen Vorgänger haben.
"20755" - <END> darf keinen Nachfolger haben.
"20756" - Externe Aufgaben dürfen keinen Vorgänger haben.
"20757" - Bei der Änderung einer externen Abhängigkeit, wurde eine ungültige Aufgabe angegeben.
"20758" - Die Angabe von externen Abhängigkeiten ist nicht möglich.

Bei ADD_TASK:

Laufende Nummer der hinzugefügten Aufgabe im Workflow
"0" - Die Aufgabe konnte nicht hinzugefügt werden.

Bei REPLACE_TASK:

Laufende Nummer der neuen Aufgabe im Workflow
"0" - Die Aufgabe konnte nicht ersetzt werden.

Start, Stop und Commit

[Start, Stop und Commit] [Workflow-Struktur ändern] [Registerkarte **Allgemein**] [Registerkarte "Frühestens"] [Registerkarte "Abhängigkeiten"] [Registerkarte **Laufzeit**] [Registerkarte "Externe Abhängigkeit"]

Syntax

MODIFY_TASK (*RunID, Status* [, **FORCED])**

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) des aktivierten Workflows Format: Script-Literal, Zahl oder Script-Variable
Status	Verarbeitungsstatus, der für die Aufgabe gesetzt werden soll Format: AE-Name, Script-Literal oder Script-Variable Erlaubte Werte: "COMMIT", "GO", "STOP" und "STOP_MODIFY" "COMMIT" - Übernimmt die Änderungen ohne den Workflow zu starten. "GO" - Startet einen gestoppten Workflow. Verwenden Sie den Parameter FORCED, um einen Workflow auch dann zu starten, wenn er im Monitor gerade zur Bearbeitung geöffnet ist. "STOP" - Stoppt einen Workflow. "STOP_MODIFY" - Stoppt einen Workflow, um ihn in weiterer Folge zu modifizieren.
FORCED	Ist in Verbindung mit dem Parameter "GO" zu verwenden. FORCED startet einen gestoppten Workflow auch dann, wenn er im Monitor gerade zur Bearbeitung geöffnet ist. Änderungen, die ein anderer Benutzer in der Zwischenzeit eventuell vorgenommen hat, gehen dabei verloren.

Anmerkungen

Die Script-Funktion ändert den Verarbeitungsstatus eines Workflows.

Beispiele

Der Workflow "MAWI.TAG" wird gestoppt, um anschließend Änderungen an ihm durchzuführen.

```
:SET&RUNID# = GET_UC_OBJECT_NR("MAWI.TAG")
:SET&RET# = MODIFY_TASK(&RUNID#, STOP_MODIFY)
```

Der Workflow "MAWI.TAG" wird gestartet.

```
:SET&RUNID# = GET_UC_OBJECT_NR("MAWI.TAG")
:SET&RET# = MODIFY_TASK(&RUNID#, GO, FORCED)
```

Workflow-Struktur ändern

[Start, Stop und Commit] [Workflow-Struktur ändern] [Registerkarte **Allgemein**] [Registerkarte "Frühestens"] [Registerkarte "Abhängigkeiten"] [Registerkarte **Laufzeit**] [Registerkarte "Externe Abhängigkeit"]

Syntax

MODIFY_TASK (*RunID, Objektname,,* ADD_TASK [, **EXTERNAL** [, *Workflow-Name]]* **)**

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) des aktivierten Workflows Format: Script-Literal, Zahl oder Script-Variable
Objektname	Name des Objektes, das in den Workflow eingefügt werden soll. Format: Script-Literal, Zahl oder Script-Variable
ADD_TASK	Fügt eine Aufgabe hinzu.
EXTERNAL	Fügt die Aufgabe als externe Abhängigkeit hinzu.
Workflow-Name	Name des Workflows auf die sich die externe Abhängigkeit beziehen soll. Format: Script-Literal oder Script-Variable

Anmerkungen

Die Script-Funktion fügt dem Workflow eine Aufgabe bzw. externe Abhängigkeit hinzu.

Die Aufgabe wird an eine freie Stelle im Workflow positioniert. Verwenden Sie MODIFY_TASK in Verbindung mit MOVE_TASK, um sie zu verschieben.

⚠ Beachten Sie, dass die hinzugefügte Aufgabe nicht mit anderen Aufgaben verbunden wird! Ziehen Sie die gewünschten Linien indem Sie MODIFY_TASK in Verbindung mit ADD_DEPENDENCY aufrufen.

⚠ Beachten Sie, dass nach dem Parameter *Objektname* zwei Kommata zu setzen sind! Beim dritten Parameter handelt es sich um die laufende Nummer einer Aufgabe innerhalb des Workflows. Er hat jedoch für das Hinzufügen einer Aufgabe keine Bedeutung.

Beispiel

Das Objekt "DATEI.EINGANG" wird dem Workflow "MAWI.TAG" hinzugefügt.

```
:SET&RUNID# = GET_UC_OBJECT_NR("MAWI.TAG")
:SET&RETSTOP# = MODIFY_TASK(&RUNID#, STOP_MODIFY)
:SET&RET# = MODIFY_TASK(&RUNID#,"DATEI.EINGANG",, ADD_TASK)
:SET&RETCOMMIT# = MODIFY_TASK(&RUNID#, COMMIT)
:SET&RETGO# = MODIFY_TASK(&RUNID#, GO)
```

Syntax

MODIFY_TASK *(RunID, [Aufgabenname], [laufende Nummer],* REPLACE_TASK *,Objektname* [, **EXTERNAL [, *Workflow-Name]]*)

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) des aktivierten Workflows Format: Script-Literal, Zahl oder Script-Variable

Aufgabenname laufende Nummer	Objektname bzw. laufende Nummer der Workflow-Aufgabe, die ersetzt werden soll. Format: Script-Literal, Zahl oder Script-Variable Eine Aufgabe in einem Workflow wird entweder mit ihrem Namen oder ihrer laufenden Nummer identifiziert. Sie haben die Möglichkeit entweder den Aufgabennamen oder die Nummer innerhalb des Workflows anzugeben. Verwenden Sie beide Parameter, so müssen der Name und die Nummer übereinstimmen. ⚠ Beachten Sie, dass bei der laufenden Nummer nicht die RunID der Aufgabe, sondern die Nummer innerhalb des Workflows gemeint ist (z.B: "3" für die dritte Aufgabe)! Die laufende Nummer ist im Aufgaben-Kästchen vermerkt. Existieren mehrere Aufgaben mit dem selben Namen im Workflow und es wird keine laufende Nummer angegeben, so wird automatisch die Aufgabe mit der niedrigsten Nummer verwendet.
REPLACE_TASK	Ersetzt eine Aufgabe.
Objektname	Name des Objektes, das die Aufgabe ersetzen soll. Format: Script-Literal oder Script-Variable
EXTERNAL	Ersetzt eine externe Abhängigkeit.
Workflow-Name	Name des Workflows auf die sich die externe Abhängigkeit bezieht. Format: Script-Literal oder Script-Variable

Anmerkungen

Die Script-Funktion ersetzt eine Workflow-Aufgabe durch ein anderes Objekt.

Eine externe Abhängigkeit kann nur durch eine externe Abhängigkeit und eine normale Workflow-Aufgabe kann nur durch eine normale Workflow-Aufgabe ersetzt werden.

Beispiel

Die externe Abhängigkeit "MAWI.DATEIEN.HOLEN" wird durch die externe Abhängigkeit "MAWI.DATEI.EINGANG" ersetzt.

```
:SET&RUNID# = GET_UC_OBJECT_NR("MAWI.TAG")
:SET&RETSTOP# = MODIFY_TASK(&RUNID#, STOP_MODIFY)
:SET&RET# = MODIFY_TASK(&RUNID#, "MAWI.DATEIEN.HOLEN",, REPLACE_TASK,
"MAWI.DATEI.EINGANG", EXTERNAL)
:SET&RETCOMMIT# = MODIFY_TASK(&RUNID#, COMMIT)
:SET&RETGO# = MODIFY_TASK(&RUNID#, GO)
```

Syntax

MODIFY_TASK*(RunID, [Aufgabenname], [laufende Nummer],* **MOVE_TASK** ***,Spalte, Zeile***)

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) des aktivierten Workflows Format: Script-Literal, Zahl oder Script-Variable
Aufgabenname *laufende Nummer*	Objektname bzw. laufende Nummer der Workflow-Aufgabe, die verschoben werden soll. Format: Script-Literal, Zahl oder Script-Variable Eine Aufgabe in einem Workflow wird entweder mit ihrem Namen oder ihrer laufenden Nummer identifiziert. Sie haben die Möglichkeit entweder den Aufgabennamen oder die Nummer innerhalb des Workflows anzugeben. Verwenden Sie beide Parameter, so müssen der Name und die Nummer übereinstimmen. ⚠ Beachten Sie, dass bei der laufenden Nummer nicht die RunID der Aufgabe, sondern die Nummer innerhalb des Workflows gemeint ist (z.B: "3" für die dritte Aufgabe)! Die laufende Nummer ist im Aufgaben-Kästchen vermerkt. ℹ Existieren mehrere Aufgaben mit dem selben Namen im Workflow und es wird keine laufende Nummer angegeben, so wird automatisch die Aufgabe mit der niedrigsten Nummer verwendet.
MOVE_TASK	Verschiebt eine Aufgabe.
Spalte	Nummer der Spalte zu der die Aufgabe verschoben werden soll. Format: Script-Literal, Zahl oder Script-Variable
Zeile	Nummer der Zeile zu der die Aufgabe verschoben werden soll. Format: Script-Literal, Zahl oder Script-Variable

Anmerkungen

Die Script-Funktion verschiebt eine Workflow-Aufgabe.

Beispiel

Die Aufgabe "DATEI.EINGANG" wird in die vierte Zeile der ersten Spalte verschoben.

```
:SET&RUNID# = GET_UC_OBJECT_NR("MAWI.TAG")
:SET&RETSTOP# = MODIFY_TASK(&RUNID#, STOP_MODIFY)
:SET&RET# = MODIFY_TASK(&RUNID#, "DATEI.EINGANG",, MOVE_TASK, 1, 4)
:SET&RETCOMMIT# = MODIFY_TASK(&RUNID#, COMMIT)
:SET&RETGO# = MODIFY_TASK(&RUNID#, GO)
```

Syntax

MODIFY_TASK*(RunID, [Aufgabenname], [laufende Nummer],* **ALIAS** *,Alias*)
MODIFY_TASK*(RunID, [Aufgabenname], [laufende Nummer],* **ALIAS_PARENT** *,Alias Parent*)

Syntaxteil	Beschreibung/Format

RunID	Laufende Nummer (RunID) des aktivierten Workflows Format: Script-Literal, Zahl oder Script-Variable
Aufgabenname *laufende Nummer*	Objektname bzw. laufende Nummer der Workflow-Aufgabe, deren Alias geändert werden soll. Format: Script-Literal, Zahl oder Script-Variable Eine Aufgabe in einem Workflow wird entweder mit ihrem Namen oder ihrer laufenden Nummer identifiziert. Sie haben die Möglichkeit entweder den Aufgabennamen oder die Nummer innerhalb des Workflows anzugeben. Verwenden Sie beide Parameter, so müssen der Name und die Nummer übereinstimmen. ⚠ Beachten Sie, dass bei der laufenden Nummer nicht die RunID der Aufgabe, sondern die Nummer innerhalb des Workflows gemeint ist (z.B: "3" für die dritte Aufgabe)! Die laufende Nummer ist im Aufgaben-Kästchen vermerkt. Existieren mehrere Aufgaben mit dem selben Namen im Workflow und es wird keine laufende Nummer angegeben, so wird automatisch die Aufgabe mit der niedrigsten Nummer verwendet.
ALIAS	Ändert den Alias einer Aufgabe oder externen Abhängigkeit im Workflow
ALIAS_PARENT	Ändert den Alias des Workflows von externen Abhängigkeiten. Nur möglich, wenn die externe Aufgabe von einem anderen Workflow stammt.
Alias Alias Parent	Alias für die Aufgabe, externe Abhängigkeit oder für den Workflow der externen Aufgabe (nur bei externen Abhängigkeiten, die auf eine Aufgabe in einem anderen Workflow verweisen) Format: AE-Name, Script-Literal, Zahl oder Script-Variable Für den Alias gelten die selben Einschränkungen wie für den Objektnamen: Maximale Länge: 200 Zeichen Erlaubte Zeichen: A-Z, 0-9, $, @, _, . und #

Anmerkungen

Die Script-Funktion ändert den Alias einer Aufgabe bzw. externen Abhängigkeit in einem Workflow.

⚠ Der Alias kann nur von Workflow-Aufgaben und externen Abhängigkeiten geändert werden, die mit dem Sprachmittel neu hinzugefügt wurden. Verwenden Sie dazu den Parameter ADD_TASK.

Wir empfehlen immer die laufende Nummer anzugeben, um die Änderung an der richtigen Aufgabe durchzuführen! Dies ist vor allem dann notwendig, wenn sich mehrere Aufgaben mit dem selben Objektnamen im Workflow befinden. Die laufende Nummer erhalten Sie als Rückgabewert der Script-Funktion, wenn Sie eine Aufgabe hinzufügen (ADD_TASK).

Beispiel

Die Aufgabe "DATEI.EINGANG" wird dem Workflow "MAWI.TAG" hinzugefügt und deren Alias auf "ALIAS.FT" geändert.

```
:SET&RUNID# = GET_UC_OBJECT_NR("MAWI.TAG")
:SET&RETSTOP# = MODIFY_TASK(&RUNID#, STOP_MODIFY)
:SET&NR# = MODIFY_TASK(&RUNID#, "DATEI.EINGANG",, ADD_TASK)
:SET&RET# = MODIFY_TASK(&RUNID#, "DATEI.EINGANG",&NR#,ALIAS,ALIAS.FT)
:SET&RETCOMMIT# = MODIFY_TASK(&RUNID#, COMMIT)
:SET&RETGO# = MODIFY_TASK(&RUNID#, GO)
```

Registerkarte Allgemein

[Start, Stop und Commit] [Workflow-Struktur ändern] [Registerkarte **Allgemein**] [Registerkarte "Frühestens"] [Registerkarte "Abhängigkeiten"] [Registerkarte **Laufzeit**] [Registerkarte "Externe Abhängigkeit"]

Syntax

MODIFY_TASK*(RunID, [Aufgabenname], [laufende Nummer],* CHECKPOINT*, Zeitstempel* *[, Zeitzone] [, Alarmobjekt])*

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) des aktivierten Objektes Format: Script-Literal, Zahl oder Script-Variable
Aufgabenname *laufende Nummer*	Name bzw. laufende Nummer der Workflow-Aufgabe. Format: Script-Literal, Zahl oder Script-Variable Eine Aufgabe in einem Workflow wird entweder mit ihrem Namen oder ihrer laufenden Nummer identifiziert. Sie haben die Möglichkeit entweder den Aufgabennamen oder die Nummer innerhalb des Workflows anzugeben. Verwenden Sie beide Parameter, so müssen der Name und die Nummer übereinstimmen. ⚠ Beachten Sie, dass bei der laufenden Nummer nicht die RunID der Aufgabe, sondern die Nummer innerhalb des Workflows gemeint ist (z.B: "3" für die dritte Aufgabe)! Die laufende Nummer ist im Aufgaben-Kästchen vermerkt.
CHECKPOINT	Setzt bzw. entfernt einen Kontrollpunkt
Zeitstempel	Zeitstempel für den Kontrollpunkt Format: Script-Literal oder Script-Variable Erlaubte Werte: Zeitstempel im Format "TT/HH:MM" - Setzt einen Kontrollpunkt mit diesem Zeitstempel "NONE" - Entfernt den Kontrollpunkt
Zeitzone	Name eines Zeitzonen-Objektes Format: Script-Literal oder Script-Variable Zeitzone für die der Zeitstempel umgerechnet werden soll.
Alarmobjekt	Objekt, das gestartet werden soll, wenn der Kontrollpunkt überschritten wird. Format: Script-Literal oder Script-Variable

Anmerkungen

Die Script-Funktion fügt einer Workflow-Aufgabe einen Kontrollpunkt hinzu oder entfernt ihn.

Beispiel

Die Aufgabe "DATEI.EINGANG" des Workflows "MAWI.TAG" erhält einen Kontrollpunkt. Das Benachrichtigungs-Objekt "TAGSCHICHT" soll gestartet werden, wenn der Kontrollpunkt überschritten wird.

```
:SET&RUNID# = GET_UC_OBJECT_NR("MAWI.TAG")
:SET&RETSTOP# = MODIFY_TASK(&RUNID#, STOP_MODIFY)
:SET&RET# = MODIFY_TASK(&RUNID#, "DATEI.EINGANG",, CHECKPOINT,
"00/15:00",,"TAGSCHICHT")
:SET&RETCOMMIT# = MODIFY_TASK(&RUNID#, COMMIT)
:SET&RETGO# = MODIFY_TASK(&RUNID#, GO)
```

Registerkarte "Frühestens"

[Start, Stop und Commit] [Workflow-Struktur ändern] [Registerkarte **Allgemein**] [Registerkarte "Frühestens"] [Registerkarte "Abhängigkeiten"] [Registerkarte **Laufzeit**] [Registerkarte "Externe Abhängigkeit"]

Syntax

MODIFY_TASK*(RunID, [Aufgabenname], [laufende Nummer],* EARLIEST_START*, Zeitstempel [, Zeitzone])*

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) des aktivierten Objektes Format: Script-Literal, Zahl oder Script-Variable
Aufgabenname *laufende Nummer*	Name bzw. laufende Nummer der Workflow-Aufgabe. Format: Script-Literal, Zahl oder Script-Variable Eine Aufgabe in einem Workflow wird entweder mit ihrem Namen oder ihrer laufenden Nummer identifiziert. Sie haben die Möglichkeit entweder den Aufgabennamen oder die Nummer innerhalb des Workflows anzugeben. Verwenden Sie beide Parameter, so müssen der Name und die Nummer übereinstimmen. ⚠ Beachten Sie, dass bei der laufenden Nummer nicht die RunID der Aufgabe, sondern die Nummer innerhalb des Workflows gemeint ist (z.B: "3" für die dritte Aufgabe)! Die laufende Nummer ist im Aufgaben-Kästchen vermerkt.
EARLIEST_START	Setzt bzw. entfernt einen frühesten Startzeitpunkt

Zeitstempel	Zeitstempel für den frühesten Startzeitpunkt Format: Script-Literal oder Script-Variable Als Vergleichswert für den frühesten Startzeitpunkt wird das reale Datum (= Generierungszeitpunkt des obersten Workflows) verwendet. Erlaubte Werte: Zeitstempel im Format "TT/HH:MM" - Setzt einen frühesten Startzeitpunkt mit diesem Zeitstempel "NONE" - Entfernt den frühesten Startzeitpunkt
Zeitzone	Name eines Zeitzonen-Objektes Format: Script-Literal oder Script-Variable Zeitzone, für die der Zeitstempel umgerechnet werden soll.

Anmerkungen

Die Script-Funktion fügt einer Workflow-Aufgabe einen frühesten Startzeitpunkt hinzu oder entfernt ihn.

Beispiel

Die Aufgabe "DATEI.EINGANG" des Workflows "MAWI.TAG" erhält einen frühesten Startzeitpunkt.

```
:SET&RUNID# = GET_UC_OBJECT_NR("MAWI.TAG")
:SET&RETSTOP# = MODIFY_TASK(&RUNID#, STOP_MODIFY)
:SET&RET# = MODIFY_TASK(&RUNID#, "DATEI.EINGANG",, EARLIEST_START,
"00/18:00")
:SET&RETCOMMIT# = MODIFY_TASK(&RUNID#, COMMIT)
:SET&RETGO# = MODIFY_TASK(&RUNID#, GO)
```

Syntax

MODIFY_TASK *(RunID, [Aufgabenname], [laufende Nummer],* ACTIVE *, Wert)*

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) des aktivierten Objektes Format: Script-Literal, Zahl oder Script-Variable

Aufgabenname *laufende Nummer*	Name bzw. laufende Nummer der Workflow-Aufgabe. Format: Script-Literal, Zahl oder Script-Variable Eine Aufgabe in einem Workflow wird entweder mit ihrem Namen oder ihrer laufenden Nummer identifiziert. Sie haben die Möglichkeit entweder den Aufgabennamen oder die Nummer innerhalb des Workflows anzugeben. Verwenden Sie beide Parameter, so müssen der Name und die Nummer übereinstimmen. ⚠ Beachten Sie, dass bei der laufenden Nummer nicht die RunID der Aufgabe, sondern die Nummer innerhalb des Workflows gemeint ist (z.B: "3" für die dritte Aufgabe)! Die laufende Nummer ist im Aufgaben-Kästchen vermerkt.
ACTIVE	Setzt eine Workflow-Aufgabe auf aktiv bzw. inaktiv.
Wert	Erlaubte Werte: "YES" und "NO" "YES" - Setzt die Workflow-Aufgabe auf aktiv. Sobald ihr Startzeitpunkt gekommen ist, wird sie durchgeführt. "NO" - Setzt die Workflow-Aufgabe auf inaktiv. Sobald ihr Startzeitpunkt gekommen ist, wird sie nicht durchgeführt.

Anmerkungen

Die Script-Funktion setzt eine Workflow-Aufgabe auf aktiv bzw. inaktiv.

Beispiel

Die Aufgabe "DATEI.EINGANG" des Workflows "MAWI.TAG" wird inaktiv gesetzt.

```
:SET&RUNID# = GET_UC_OBJECT_NR("MAWI.TAG")
:SET&RETSTOP# = MODIFY_TASK(&RUNID#, STOP_MODIFY)
:SET&RET# = MODIFY_TASK(&RUNID#, "DATEI.EINGANG",, ACTIVE, "NO")
:SET&RETCOMMIT# = MODIFY_TASK(&RUNID#, COMMIT)
:SET&RETGO# = MODIFY_TASK(&RUNID#, GO)
```

Syntax

MODIFY_TASK *(RunID, [Aufgabenname], [laufende Nummer],* BREAKPOINT*, Wert)*

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) des aktivierten Objektes Format: Script-Literal, Zahl oder Script-Variable

Aufgabenname *laufende Nummer*	Name bzw. laufende Nummer der Workflow-Aufgabe. Format: Script-Literal, Zahl oder Script-Variable Eine Aufgabe in einem Workflow wird entweder mit ihrem Namen oder ihrer laufenden Nummer identifiziert. Sie haben die Möglichkeit entweder den Aufgabennamen oder die Nummer innerhalb des Workflows anzugeben. Verwenden Sie beide Parameter, so müssen der Name und die Nummer übereinstimmen. ⚠ Beachten Sie, dass bei der laufenden Nummer nicht die RunID der Aufgabe, sondern die Nummer innerhalb des Workflows gemeint ist (z.B: "3" für die dritte Aufgabe)! Die laufende Nummer ist im Aufgaben-Kästchen vermerkt.
BREAKPOINT	Setzt bzw. entfernt einen Haltepunkt auf eine Workflow-Aufgabe.
Wert	Erlaubte Werte: "YES" und "NO" "YES" - Setzt einen Haltepunkt bei dieser Workflow-Aufgabe. "NO" - Entfernt den Haltepunkt bei dieser Workflow-Aufgabe.

Anmerkungen

Die Script-Funktion setzt bzw. entfernt einen Haltepunkt bei einer Workflow-Aufgabe.

Beispiel

Bei der Aufgabe "DATEI.EINGANG" des Workflows "MAWI.TAG" wird ein Haltepunkt gesetzt.

```
:SET&RUNID# = GET_UC_OBJECT_NR("MAWI.TAG")
:SET&RETSTOP# = MODIFY_TASK(&RUNID#, STOP_MODIFY)
:SET&RET# = MODIFY_TASK(&RUNID#, "DATEI.EINGANG",, BREAKPOINT, "YES")
:SET&RETCOMMIT# = MODIFY_TASK(&RUNID#, COMMIT)
:SET&RETGO# = MODIFY_TASK(&RUNID#, GO)
```

Registerkarte "Abhängigkeiten"

[Start, Stop und Commit] [Workflow-Struktur ändern] [Registerkarte **Allgemein**] [Registerkarte "Frühestens"] [Registerkarte "Abhängigkeiten"] [Registerkarte **Laufzeit**] [Registerkarte "Externe Abhängigkeit"]

Syntax

MODIFY_TASK*(RunID, [Aufgabenname], [laufende Nummer],* LATEST_START*, Zeitstempel *[, Zeitzone]*)

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) des aktivierten Objektes Format: Script-Literal, Zahl oder Script-Variable

Aufgabenname *laufende Nummer*	Name bzw. laufende Nummer der Workflow-Aufgabe. Format: Script-Literal, Zahl oder Script-Variable Eine Aufgabe in einem Workflow wird entweder mit ihrem Namen oder ihrer laufenden Nummer identifiziert. Sie haben die Möglichkeit entweder den Aufgabennamen oder die Nummer innerhalb des Workflows anzugeben. Verwenden Sie beide Parameter, so müssen der Name und die Nummer übereinstimmen. ⚠ Beachten Sie, dass bei der laufenden Nummer nicht die RunID der Aufgabe, sondern die Nummer innerhalb des Workflows gemeint ist (z.B: "3" für die dritte Aufgabe)! Die laufende Nummer ist im Aufgaben-Kästchen vermerkt.
LATEST_START	Setzt bzw. entfernt einen spätesten Startzeitpunkt
Zeitstempel	Zeitstempel für den spätesten Startzeitpunkt Format: Script-Literal oder Script-Variable Erlaubte Werte: Zeitstempel im Format "TT/HH:MM" - Setzt einen spätesten Startzeitpunkt mit diesem Zeitstempel "NONE" - Entfernt den spätesten Startzeitpunkt
Zeitzone	Name eines Zeitzonen-Objektes Format: Script-Literal oder Script-Variable Zeitzone, für die der Zeitstempel umgerechnet werden soll.

Anmerkungen

Die Script-Funktion fügt einer Workflow-Aufgabe einen spätesten Startzeitpunkt hinzu oder entfernt ihn.

Beispiel

Die Aufgabe "DATEI.EINGANG" des Workflows "MAWI.TAG" erhält einen spätesten Startzeitpunkt.

```
:SET&RUNID# = GET_UC_OBJECT_NR("MAWI.TAG")
:SET&RETSTOP# = MODIFY_TASK(&RUNID#, STOP_MODIFY)
:SET&RET# = MODIFY_TASK(&RUNID#, "DATEI.EINGANG",, LATEST_START,
"00/10:00")
:SET&RETCOMMIT# = MODIFY_TASK(&RUNID#, COMMIT)
:SET&RETGO# = MODIFY_TASK(&RUNID#, GO)
```

Syntax

MODIFY_TASK*(RunID, [Aufgabenname], [laufende Nummer],* LATEST_END*, Zeitstempel [, Zeitzone]*)

Syntaxteil	Beschreibung/Format

RunID	Laufende Nummer (RunID) des aktivierten Objektes Format: Script-Literal, Zahl oder Script-Variable
Aufgabenname *laufende Nummer*	Name bzw. laufende Nummer der Workflow-Aufgabe. Format: Script-Literal, Zahl oder Script-Variable Eine Aufgabe in einem Workflow wird entweder mit ihrem Namen oder ihrer laufenden Nummer identifiziert. Sie haben die Möglichkeit entweder den Aufgabennamen oder die Nummer innerhalb des Workflows anzugeben. Verwenden Sie beide Parameter, so müssen der Name und die Nummer übereinstimmen. ⚠ Beachten Sie, dass bei der laufenden Nummer nicht die RunID der Aufgabe, sondern die Nummer innerhalb des Workflows gemeint ist (z.B: "3" für die dritte Aufgabe)! Die laufende Nummer ist im Aufgaben-Kästchen vermerkt.
LATEST_END	Setzt bzw. entfernt einen spätesten Endezeitpunkt
Zeitstempel	Zeitstempel für den spätesten Endezeitpunkt Format: Script-Literal oder Script-Variable Erlaubte Werte: Zeitstempel im Format "TT/HH:MM" - Setzt einen spätesten Endezeitpunkt mit diesem Zeitstempel "NONE" - Entfernt den spätesten Endezeitpunkt
Zeitzone	Name eines Zeitzonen-Objektes Format: Script-Literal oder Script-Variable Zeitzone, für die der Zeitstempel umgerechnet werden soll.

Anmerkungen

Die Script-Funktion fügt einer Workflow-Aufgabe einen spätesten Endezeitpunkt hinzu oder entfernt ihn.

Beispiel

Die Aufgabe "DATEI.EINGANG" des Workflows "MAWI.TAG" erhält einen spätesten Endezeitpunkt.

```
:SET&RUNID# = GET_UC_OBJECT_NR("MAWI.TAG")
:SET&RETSTOP# = MODIFY_TASK(&RUNID#, STOP_MODIFY)
:SET&RET# = MODIFY_TASK(&RUNID#, "DATEI.EINGANG",, LATEST_END,
"00/18:00")
:SET&RETCOMMIT# = MODIFY_TASK(&RUNID#, COMMIT)
:SET&RETGO# = MODIFY_TASK(&RUNID#, GO)
```

Syntax

MODIFY_TASK*(RunID, [Aufgabenname], [laufende Nummer],* DEPENDENCY_STATE_MATCH *, Wert*)

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) des aktivierten Objektes Format: Script-Literal, Zahl oder Script-Variable
Aufgabenname *laufende Nummer*	Name bzw. laufende Nummer der Workflow-Aufgabe. Format: Script-Literal, Zahl oder Script-Variable Eine Aufgabe in einem Workflow wird entweder mit ihrem Namen oder ihrer laufenden Nummer identifiziert. Sie haben die Möglichkeit entweder den Aufgabennamen oder die Nummer innerhalb des Workflows anzugeben. Verwenden Sie beide Parameter, so müssen der Name und die Nummer übereinstimmen. ⚠ Beachten Sie, dass bei der laufenden Nummer nicht die RunID der Aufgabe, sondern die Nummer innerhalb des Workflows gemeint ist (z.B: "3" für die dritte Aufgabe)! Die laufende Nummer ist im Aufgaben-Kästchen vermerkt.
DEPENDENCY_STATE_MATCH	Beeinflusst wieviele Status der Vorgänger zutreffen müssen
Wert	Erlaubte Werte. "ALL" und "ONE" "ALL" - Alle Status müssen zutreffen "ONE" - Mindestens 1 Status muss zutreffen

Anmerkungen

Die Script-Funktion bestimmt wieviele Status der Vorgänger zutreffen müssen, damit die Workflow-Aufgabe durchgeführt werden kann.

Beispiel

Alle Vorgänger der Aufgabe "DATEI.EINGANG" des Workflows "MAWI.TAG" müssen dem erwarteten Status entsprechen.

```
:SET&RUNID# = GET_UC_OBJECT_NR("MAWI.TAG")
:SET&RETSTOP# = MODIFY_TASK(&RUNID#, STOP_MODIFY)
:SET&RET# = MODIFY_TASK(&RUNID#, "DATEI.EINGANG",, DEPENDENCY_STATE_
MATCH, "ALL")
:SET&RETCOMMIT# = MODIFY_TASK(&RUNID#, COMMIT)
:SET&RETGO# = MODIFY_TASK(&RUNID#, GO)
```

Syntax

MODIFY_TASK *(RunID, [Nachfolger-Aufgabenname], [laufende Nummer], Abhängigkeit,[Quell-Aufgabenname] , [laufende Nummer],*

***Status*)**

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) des aktivierten Workflows Format: Script-Literal, Zahl oder Script-Variable
Nachfolger-Aufgabenname laufende Nummer	Name bzw. laufende Nummer der Nachfolger-Aufgabe. Format: Script-Literal, Zahl oder Script-Variable Eine Aufgabe in einem Workflow wird entweder mit ihrem Namen oder ihrer laufenden Nummer identifiziert. Sie haben die Möglichkeit entweder den Aufgabennamen oder die Nummer innerhalb des Workflows anzugeben. Verwenden Sie beide Parameter, so müssen der Name und die Nummer übereinstimmen. ⚠ Beachten Sie, dass bei der laufenden Nummer nicht die RunID der Aufgabe, sondern die Nummer innerhalb des Workflows gemeint ist (z.B: "3" für die dritte Aufgabe)! Die laufende Nummer ist im Aufgaben-Kästchen vermerkt.
Abhängigkeit	Aktion zum Bearbeiten der Abhängigkeiten Erlaubte Werte: "ADD_DEPENDENCY", "MODIFY_DEPENDENCY" und "REMOVE_DEPENDENCY" "ADD_DEPENDENCY" - fügt eine Abhängigkeit hinzu "MODIFY_DEPENDENCY" - ändert eine Abhängigkeit "REMOVE_DEPENDENCY" - entfernt eine Abhängigkeit
Aufgabenname laufende Nummer	Name bzw. laufende Nummer der Aufgabe bei der die Abhängigkeit geändert werden soll Format: Script-Literal, Zahl oder Script-Variable Eine Aufgabe in einem Workflow wird entweder mit ihrem Namen oder ihrer laufenden Nummer identifiziert. Sie haben die Möglichkeit entweder den Aufgabennamen oder die Nummer innerhalb des Workflows anzugeben. Verwenden Sie beide Parameter, so müssen der Name und die Nummer übereinstimmen. ⚠ Beachten Sie, dass bei der laufenden Nummer nicht die RunID der Aufgabe, sondern die Nummer innerhalb des Workflows gemeint ist (z.B: "3" für die dritte Aufgabe)! Die laufende Nummer ist im Aufgaben-Kästchen vermerkt.
Status	Status, der für die Vorgängeraufgabe bei Durchführungsende erwartet wird. Format: Script-Literal oder Script-Variable Erlaubte Werte: "*Endestatus*" und "NONE" "*Endestatus*" - z.B: ENDED_OK "NONE" - Der Endestatus der Vorgängeraufgabe wird nicht überwacht. ℹ Bei "REMOVE_DEPENDENCY" muss kein Status angegeben werden.

Anmerkungen

Die Script-Funktion fügt einer Workflow-Aufgabe eine Abhängigkeit hinzu, ändert oder entfernt sie. Sie verbindet damit auch Aufgaben durch Linien.

Beispiel

Das START-Kästchen wird mit der 4. Aufgabe verbunden. Zusätzlich ändert sich der Ende-Status, den die 9. Aufgabe von seinem Vorgänger der 8. Aufgabe erwartet, auf "ANY_OK". Die Abhängigkeit und damit auch die Linie zwischen der 10. und 11. Aufgabe wird entfernt.

```
:SET&RUNID# = GET_UC_OBJECT_NR("MAWI.TAG")
:SET&RETSTOP# = MODIFY_TASK(&RUNID#, STOP_MODIFY)
:SET&RET# = MODIFY_TASK(&RUNID#,, 4, ADD_DEPENDENCY, "START",, "ENDED_
OK")
:SET&RET# = MODIFY_TASK(&RUNID#,, 9, MODIFY_DEPENDENCY,, 8,"ANY_OK")
:SET&RET# = MODIFY_TASK(&RUNID#,, 11, REMOVE_DEPENDENCY,, 10, "ENDED_
OK")
:SET&RETCOMMIT# = MODIFY_TASK(&RUNID#, COMMIT)
:SET&RETGO# = MODIFY_TASK(&RUNID#, GO)
```

Syntax

MODIFY_TASK*(RunID, [Aufgabenname], [laufende Nummer],* DEPENDENCY_ELSE_ACTION *, Sonst-Aktion*[*, Alarmobjekt*])

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) des aktivierten Objektes Format: Script-Literal, Zahl oder Script-Variable
Aufgabenname *laufende Nummer*	Name bzw. laufende Nummer der Workflow-Aufgabe. Format: Script-Literal, Zahl oder Script-Variable Eine Aufgabe in einem Workflow wird entweder mit ihrem Namen oder ihrer laufenden Nummer identifiziert. Sie haben die Möglichkeit entweder den Aufgabennamen oder die Nummer innerhalb des Workflows anzugeben. Verwenden Sie beide Parameter, so müssen der Name und die Nummer übereinstimmen. ⚠ Beachten Sie, dass bei der laufenden Nummer nicht die RunID der Aufgabe, sondern die Nummer innerhalb des Workflows gemeint ist (z.B: "3" für die dritte Aufgabe)! Die laufende Nummer ist im Aufgaben-Kästchen vermerkt.
DEPENDENCY_ELSE_ACTION	Bestimmt die Aktion, die ausgeführt werden soll, wenn die Abhängigkeiten nicht wie erwartet eintreffen.

Sonst-Aktion	Erlaubte Werte: "ABORT", "BLOCK", "BLOCK_ABORT" und "SKIP" "ABORT" - Die Aufgabe und der Workflow selbst werden abgebrochen. "BLOCK" - Der Workflow blockiert bei der Vorgängeraufgabe. "BLOCK_ABORT" - Der Workflow blockiert bei der Vorgängeraufgabe und schickt ein Abbruch-Signal an den übergeordneten Workflow (sofern vorhanden). "SKIP" - Die Ablauf-Aufgabe wird übersprungen.
Alarmobjekt	Objekt, das gestartet werden soll, wenn der Kontrollpunkt überschritten wird. Format: Script-Literal oder Script-Variable

Anmerkungen

Die Script-Funktion bestimmt die Aktion, die ausgeführt werden soll, wenn die Abhängigkeiten nicht wie erwartet eintreffen.

Beispiel

Die Aufgabe "DATEI.EINGANG" des Workflows "MAWI.TAG" wird übersprungen, wenn die Abhängigkeiten nicht zutreffen.

```
:SET&RUNID# = GET_UC_OBJECT_NR("MAWI.TAG")
:SET&RETSTOP# = MODIFY_TASK(&RUNID#, STOP_MODIFY)
:SET&RET# = MODIFY_TASK(&RUNID#, "DATEI.EINGANG",, DEPENDENCY_ELSE_
ACTION, "SKIP")
:SET&RETCOMMIT# = MODIFY_TASK(&RUNID#, COMMIT)
:SET&RETGO# = MODIFY_TASK(&RUNID#, GO)
```

Registerkarte Laufzeit

[Start, Stop und Commit] [Workflow-Struktur ändern] [Registerkarte **Allgemein**] [Registerkarte "Frühestens"] [Registerkarte "Abhängigkeiten"] [Registerkarte **Laufzeit**] [Registerkarte "Externe Abhängigkeit"]

Syntax

MODIFY_TASK*(RunID, [Aufgabenname], [laufende Nummer], Laufzeitoption, Wert*)

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) des aktivierten Objektes Format: Script-Literal, Zahl oder Script-Variable

Aufgabenname laufende Nummer	Name bzw. laufende Nummer der Workflow-Aufgabe. Format: Script-Literal, Zahl oder Script-Variable Eine Aufgabe in einem Workflow wird entweder mit ihrem Namen oder ihrer laufenden Nummer identifiziert. Sie haben die Möglichkeit entweder den Aufgabennamen oder die Nummer innerhalb des Workflows anzugeben. Verwenden Sie beide Parameter, so müssen der Name und die Nummer übereinstimmen. ⚠ Beachten Sie, dass bei der laufenden Nummer nicht die RunID der Aufgabe, sondern die Nummer innerhalb des Workflows gemeint ist (z.B: "3" für die dritte Aufgabe)! Die laufende Nummer ist im Aufgaben-Kästchen vermerkt.
Laufzeitoption	Laufzeitoption, die geändert werden soll. Erlaubte Werte: "RUNTIME_USE_TASK_SETTINGS" - MRT/SRT-Einstellungen der Aufgabe verwenden "RUNTIME_MRT_NONE" - Keine maximale Laufzeit "RUNTIME_MRT_FIXED" - Fixwert für die maximale Laufzeit "RUNTIME_MRT_ERT" - ERT plus Prozentsatz für die maximale Laufzeit "RUNTIME_MRT_TIME" - Aktuelles Datum plus Zeitraum für die maximale Laufzeit "RUNTIME_SRT_NONE" - Keine minimale Laufzeit "RUNTIME_SRT_FIXED" - Fixwert für die minimale Laufzeit "RUNTIME_SRT_ERT" - ERT minus Prozentsatz für die minimale Laufzeit "RUNTIME_ELSE_ACTION" - Sonst-Aktion, wenn die Laufzeit abweicht

Wert	Der Wert richtet sich nach der Laufzeitoption: Für "RUNTIME_USE_TASK_SETTINGS": "YES" und "NO" "YES" - Laufzeitoptionen der Aufgabe verwenden "NO" - Laufzeitoptionen der Eigenschaften verwenden Für "RUNTIME_MRT_NONE": Kein Wert Für "RUNTIME_MRT_FIXED": Fixwert im Format "HHHH:MM:SS" Für "RUNTIME_MRT_ERT": Prozentsatz Für "RUNTIME_MRT_TIME": "TT/HH:MM" *[, Zeitzone]* "TT" - Tage, die zum aktuellen Datum addiert werden "HH:MM" - Uhrzeit Zeitzone in die der Zeitstempel umgerechnet werden soll Für "RUNTIME_SRT_NONE": Kein Wert Für "RUNTIME_SRT_FIXED": Fixwert im Format "HHHH:MM:SS" Für "RUNTIME_SRT_ERT": Prozentsatz Für "RUNTIME_ELSE_ACTION": *Sonst-Aktion [, Alarmobjekt]* Sonst-Aktion - "CANCEL" (Aufgabe abbrechen/beenden) und "NONE" (keine Sonst-Aktion ausführen) Alarmobjekt, das gestartet werden soll, wenn die erwartete Laufzeit überschritten wird.

Anmerkungen

Die Script-Funktion ändert die Laufzeit-Einstellungen einer Workflow-Aufgabe.

Beispiel

Dauert die Durchführung der Aufgabe "DATEI.EINGANG" länger als zwei Stunden, wird das Benachrichtigungsobjekt "TAGSCHICHT" gestartet. Die minimale Laufzeit ist unerheblich.

```
:SET&RUNID# = GET_UC_OBJECT_NR("MAWI.TAG")
:SET&RETSTOP# = MODIFY_TASK(&RUNID#, STOP_MODIFY)
:SET&RETMRT# = MODIFY_TASK(&RUNID#, "DATEI.EINGANG",, RUNTIME_MRT_
FIXED, "2:00:00")
:SET &RETSRT# = MODIFY_TASK(&RUNID#, "DATEI.EINGANG",, RUNTIME_SRT_
NONE)
:SET &RETACT# = MODIFY_TASK(&RUNID#, "DATEI.EINGANG",, RUNTIME_ELSE_
ACTION,"CANCEL", "TAGSCHICHT")
:SET&RETCOMMIT# = MODIFY_TASK(&RUNID#, COMMIT)
:SET&RETGO# = MODIFY_TASK(&RUNID#, GO)
```

Registerkarte "Externe Abhängigkeit"

[Start, Stop und Commit] [Workflow-Struktur ändern] [Registerkarte **Allgemein**] [Registerkarte "Frühestens"] [Registerkarte "Abhängigkeiten"] [Registerkarte **Laufzeit**] [Registerkarte "Externe Abhängigkeit"]

Syntax

MODIFY_TASK*(RunID, [Aufgabenname], [laufende Nummer],*EXTERNAL_SATISFACTION *, Erfüllungszeitraum [, Dauer]*)

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) des aktivierten Objektes Format: Script-Literal, Zahl oder Script-Variable
Aufgabenname *laufende Nummer*	Name bzw. laufende Nummer der Workflow-Aufgabe. Format: Script-Literal, Zahl oder Script-Variable Eine Aufgabe in einem Workflow wird entweder mit ihrem Namen oder ihrer laufenden Nummer identifiziert. Sie haben die Möglichkeit entweder den Aufgabennamen oder die Nummer innerhalb des Workflows anzugeben. Verwenden Sie beide Parameter, so müssen der Name und die Nummer übereinstimmen. ⚠ Beachten Sie, dass bei der laufenden Nummer nicht die RunID der Aufgabe, sondern die Nummer innerhalb des Workflows gemeint ist (z.B: "3" für die dritte Aufgabe)! Die laufende Nummer ist im Aufgaben-Kästchen vermerkt.
EXTERNAL_SATISFACTION	Legt den Erfüllungszeitraum fest.
Erfüllungszeitraum	Gültigkeit des Erfüllungszeitraumes Format: Script-Literal oder Script-Variable Erlaubte Werte: "LOGICAL_DATE", "LAST_EXECUTION", "BEFORE_START" und "AFTER_START" "LOGICAL_DATE" - Nur gültig, wenn mit gleichem logischem Datum aktiviert "LAST_EXECUTION" - Seit letzter Workflow-Durchführung "BEFORE_START" - Innerhalb eines Zeitraumes vor Workflow-Start "AFTER_START" - Nach Workflow-Start
Dauer	Zeitraum für die Option "BEFORE_START" im Format "HH:MM:SS" Format: Script-Literal oder Script-Variable

Anmerkungen

Die Script-Funktion legt den Erfüllungszeitraum einer externen Abhängigkeit fest.

Beispiel

Der Erfüllungszeitraum für die externe Abhängigkeit "MAWI.DATEIEN.HOLEN" beginnt mit dem Start des Workflows.

```
:SET&RUNID# = GET_UC_OBJECT_NR("MAWI.TAG")
:SET&RETSTOP# = MODIFY_TASK(&RUNID#, STOP_MODIFY)
:SET&RET# = MODIFY_TASK(&RUNID#, "MAWI.DATEIEN.HOLEN",, EXTERNAL_
SATISFACTION, "AFTER_START")
:SET&RETCOMMIT# = MODIFY_TASK(&RUNID#, COMMIT)
:SET&RETGO# = MODIFY_TASK(&RUNID#, GO)
```

Syntax

MODIFY_TASK*(RunID, [Aufgabenname], [laufende Nummer],* EXTERNAL_STATUS *, Status*)

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) des aktivierten Objektes Format: Script-Literal, Zahl oder Script-Variable
Aufgabenname *laufende Nummer*	Name bzw. laufende Nummer der Workflow-Aufgabe. Format: Script-Literal, Zahl oder Script-Variable Eine Aufgabe in einem Workflow wird entweder mit ihrem Namen oder ihrer laufenden Nummer identifiziert. Sie haben die Möglichkeit entweder den Aufgabennamen oder die Nummer innerhalb des Workflows anzugeben. Verwenden Sie beide Parameter, so müssen der Name und die Nummer übereinstimmen. ⚠ Beachten Sie, dass bei der laufenden Nummer nicht die RunID der Aufgabe, sondern die Nummer innerhalb des Workflows gemeint ist (z.B: "3" für die dritte Aufgabe)! Die laufende Nummer ist im Aufgaben-Kästchen vermerkt.
EXTERNAL_STATUS	Setzt bzw. entfernt den erwarteten Status.
Status	Status, der für die externe Aufgabe bei Durchführungsende erwartet wird. Format: Script-Literal oder Script-Variable Erlaubte Werte: "*Endestatus*" und "NONE" "*Endestatus*" - z.B: ENDED_OK "NONE" - Es wird nur geprüft, ob die Aufgabe im Erfüllungszeitraum endete. Ihr Status spielt in diesem Fall keine Rolle.

Anmerkungen

Die Script-Funktion setzt den erwarteten Status einer externen Abhängigkeit.

Beispiel

Der Workflow "MAWI.TAG" erwartet den Status "ENDED_OK" für die externe Abhängigkeit auf "MAWI.DATEIEN.HOLEN".

```
:SET&RUNID# = GET_UC_OBJECT_NR("MAWI.TAG")
:SET&RETSTOP# = MODIFY_TASK(&RUNID#, STOP_MODIFY)
:SET&RET# = MODIFY_TASK(&RUNID#, "MAWI.DATEIEN.HOLEN",, EXTERNAL_
STATUS, "ENDED_OK")
:SET&RETCOMMIT# = MODIFY_TASK(&RUNID#, COMMIT)
:SET&RETGO# = MODIFY_TASK(&RUNID#, GO)
```

Syntax

MODIFY_TASK*(RunID, [Aufgabenname], [laufende Nummer],* EXTERNAL_ELSE_ACTION *, Aktion* *[, Alarmobjekt]*)

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) des aktivierten Objektes Format: Script-Literal, Zahl oder Script-Variable
Aufgabenname *laufende Nummer*	Name bzw. laufende Nummer der Workflow-Aufgabe. Format: Script-Literal, Zahl oder Script-Variable Eine Aufgabe in einem Workflow wird entweder mit ihrem Namen oder ihrer laufenden Nummer identifiziert. Sie haben die Möglichkeit entweder den Aufgabennamen oder die Nummer innerhalb des Workflows anzugeben. Verwenden Sie beide Parameter, so müssen der Name und die Nummer übereinstimmen. ⚠ Beachten Sie, dass bei der laufenden Nummer nicht die RunID der Aufgabe, sondern die Nummer innerhalb des Workflows gemeint ist (z.B: "3" für die dritte Aufgabe)! Die laufende Nummer ist im Aufgaben-Kästchen vermerkt.
EXTERNAL_ELSE_ACTION	Bestimmt die Aktion, die ausgeführt werden soll, wenn die externe Aufgabe im Erfüllungszeitraum entweder nicht den erwarteten Status mindestens einmal erreicht hat oder gar nicht endete.
Aktion	Erlaubte Werte: "WAIT", "SKIP" und "CANCEL" "WAIT" - Warten "SKIP" - Überspringen "CANCEL" - Workflow abbrechen
Alarmobjekt	Objekt, das gestartet werden soll, wenn ein Timeout eintritt. Format: Script-Literal oder Script-Variable

Anmerkungen

Die Script-Funktion bestimmt die Sonst-Aktion einer externen Abhängigkeit.

Beispiel

Der Workflow "MAWI.TAG wird abgebrochen, wenn die externe Abhängigkeit auf "MAWI.DATEIEN.HOLEN" nicht wie erwartet eintrifft. Das Benachrichtigungs-Objekt "TAGSCHICHT" soll in diesem Fall gestartet werden.

```
:SET&RUNID# = GET_UC_OBJECT_NR("MAWI.TAG")
:SET&RETSTOP# = MODIFY_TASK(&RUNID#, STOP_MODIFY)
:SET&RET# = MODIFY_TASK(&RUNID#, "MAWI.DATEIEN.HOLEN",, EXTERNAL_ELSE_
ACTION, "CANCEL", "TAGSCHICHT")
:SET&RETCOMMIT# = MODIFY_TASK(&RUNID#, COMMIT)
:SET&RETGO# = MODIFY_TASK(&RUNID#, GO)
```

Syntax

MODIFY_TASK*(RunID, [Aufgabenname], [laufende Nummer],* **EXTERNAL_TIMEOUT_ACTION** *,* ***Timeoutlänge*** **, Aktion** *[, Alarmobjekt]*)

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) des aktivierten Objektes Format: Script-Literal, Zahl oder Script-Variable
Aufgabenname *laufende Nummer*	Name bzw. laufende Nummer der Workflow-Aufgabe. Format: Script-Literal, Zahl oder Script-Variable Eine Aufgabe in einem Workflow wird entweder mit ihrem Namen oder ihrer laufenden Nummer identifiziert. Sie haben die Möglichkeit entweder den Aufgabennamen oder die Nummer innerhalb des Workflows anzugeben. Verwenden Sie beide Parameter, so müssen der Name und die Nummer übereinstimmen. ⚠ Beachten Sie, dass bei der laufenden Nummer nicht die RunID der Aufgabe, sondern die Nummer innerhalb des Workflows gemeint ist (z.B: "3" für die dritte Aufgabe)! Die laufende Nummer ist im Aufgaben-Kästchen vermerkt.
EXTERNAL_TIMEOUT_ACTION	Legt die Zeitspanne fest, die bei der Sonst-Aktion "WAIT" abgewartet werden soll.
Timeoutlänge	Dauer des Timeouts im Format "HH:MM:SS" Format: Script-Literal oder Script-Variable
Aktion	Erlaubte Werte: "WAIT", "SKIP" und "CANCEL" "WAIT" - Warten "SKIP" - Überspringen "CANCEL" - Workflow abbrechen
Alarmobjekt	Objekt, das gestartet werden soll, wenn ein Timeout eintritt. Format: Script-Literal oder Script-Variable

Anmerkungen

Die Script-Funktion legt die Timeout-Einstellung einer externen Abhängigkeit fest.

Beispiel

Die externe Abhängigkeit auf "MAWI.DATEIEN.HOLEN" wird übersprungen, wenn eine Stunde abgelaufen ist.

```
:SET&RUNID# = GET_UC_OBJECT_NR("MAWI.TAG")
:SET&RETSTOP# = MODIFY_TASK(&RUNID#, STOP_MODIFY)
:SET&RET# = MODIFY_TASK(&RUNID#, "MAWI.DATEIEN.HOLEN",, EXTERNAL_
TIMEOUT_ACTION, "01:00:00", "SKIP")
:SET&RETCOMMIT# = MODIFY_TASK(&RUNID#, COMMIT)
:SET&RETGO# = MODIFY_TASK(&RUNID#, GO)
```

Siehe auch:

Script-Sprachmittel	Beschreibung
MODIFY_UC_OBJECT	Ändert das Attribut eines aktivierten Objektes.

Script-Sprachmittel - Objekte aktivieren
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.28 MODIFY_UC_OBJECT

Script-Funktion: Ändert das Attribut eines aktivierten Objektes.

Syntax

MODIFY_UC_OBJECT*(RunID, Attribut, Wert)*

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) des aktivierten Objektes Format: Script-Literal, Zahl oder Script-Variable Bei der Verwendung des Attributes EARLIEST_STARTTIME ist nicht die RunID der Aufgabe, sondern jene des Workflows in dem sich die Aufgabe befindet anzugeben.
Attribut	Attribut, welches geändert werden soll. Format: AE-Name oder Script-Variable
Wert	Wert, der für das Attribut gesetzt wird. Format: AE-Name, Script-Literal, Zahl oder Script-Variable

Rückgabewerte

"0" - Die Änderung des Attributes war erfolgreich.
"20342" - Das Attribut wird nicht unterstützt.

Für das Attribut GAP gilt:
20534 - Der Wert "0" ist nicht erlaubt.
20591 - Die RunID wurde nicht gefunden.
20860 - Es handelt sich nicht um eine periodische Aufgabe.
20861 - Bei der periodischen Aufgabe wurde nicht die Option "Im Abstand von..." ausgewählt.

Für die Attribute GOIMM, EARLIEST_STARTTIME und REMOVE_DEPENDENCY gelten:
"20378" - Der Workflow wurde nicht gefunden.
"20840" - Die Aufgabe kommt mehrfach im Workflow vor.
"20841" - Die angegebene Aufgabe existiert im Workflow nicht.

Für das Attribut GR_MAX_PAR_JOBS gelten:
"11121" - Die RunID wurde nicht gefunden.
"20460" - Der Wert liegt nicht zwischen 1 und 999.

Für das Attribut IGNORE_SYNC gilt:
"11191" - Die RunID wurde nicht gefunden.

Für das Attribut PRIORITY gelten:
"20591" - Die RunID wurde nicht gefunden.
"20587" - Der Wert liegt nicht zwischen 0 und 255.

Für das Attribut RELEASE gelten:
"11016" - Der zu ändernde Workflow existiert nicht.
"11510" - Der Workflow ist nicht aktiv.
"20540" - Die RunID wurde nicht gefunden.

Für das Attribut RESPONSE gelten:
"20539" - Die Benachrichtigung mit der angegebenen RunID existiert nicht.
"20538" - Der Wert ist nicht korrekt.
"11151" - Der Wert für das Attribut RESPONSE trifft nicht für den Typ der Benachrichtigung (Benachrichtigung, Rückfrage, Alarm) zu.
"11060" - Die Kombination Benachrichtigung/Wert wird nicht unterstützt.
"11061" - Die Benachrichtigung wurde schon von einem anderen Benutzer akzeptiert.
"11062" - Das Bestätigen der Benachrichtigung kommt von einem Benutzer, der nicht im Benachrichtigungs-Objekt vorhanden ist.
"11064" - Die Ablehnung der Benachrichtigung kommt von einem Benutzer, der nicht im Benachrichtigungs-Objekt vorhanden ist.
"20859" - Die Reaktion auf die Benachrichtigung kommt von einem Benutzer, der inaktiv im Benachrichtigungs-Objekt vorhanden ist.

Anmerkungen

Es können die in der Tabelle genannten Attribute von aktivierten Objekten geändert werden. Die neuen Werte gelten für die aktuelle Durchführung der Aufgabe und werden daher nicht im Objekt gespeichert.

Objekttyp	Attribut	Wert

CALL	RESPONSE Reaktion auf einer Benachrichtigung.	"OK" = Benachrichtigung (Benachrichtigung) quittieren. "YES" = Benachrichtigung (Rückfrage) mit "Ja" beantworten. "NO" = Benachrichtigung (Rückfrage) mit "Nein" beantworten. "ACCEPT" = Benachrichtigung (Alarmierung) akzeptieren. "REJECT" = Benachrichtigung (Alarmierung) ablehnen. "DONE" = Benachrichtigung (Alarmierung) erledigen.
JOBG	GR_MAX_PAR_JOBS Maximale Anzahl parallel laufende Aufgaben in einer Gruppe.	1 bis 999 parallel laufende Aufgaben.
JOBP	RELEASE Freigabe eines blockierten Workflows.	Laufende Nummer (RunID) der im Workflow blockierten Aufgabe.
	Für die folgenden drei Attribute muss sich die zu ändernde Aufgabe im Zustand "Warten auf Vorbedingung" befinden.	
	GOIMM Die Aufgabe eines Workflows wird sofort gestartet	Name der Aufgabe. Diese darf im Workflow nur einmal vorkommen.

	EARLIEST_STARTTIME Änderung der frühesten Startzeit einer Aufgabe in einem Workflow	Geben Sie entweder eine früheste Startzeit oder das Schlüsselwort "OFF" an. In beiden Fällen benötigt die Script-Funktion auch den Namen der Aufgabe. Syntax für den Parameter *Wert*: *Name der Aufgabe, [Zeitformat;]Zeit* bzw. *Name der Aufgabe,* OFF Die Startzeit wird der Aufgabe zugewiesen bzw. ersetzt eine bereits gesetzte. Die Script-Funktion erwartet das Zeitformat "HHMMSS", wenn Sie kein spezielles definieren. Das Schlüsselwort "OFF" entfernt eine vorhandene früheste Startzeit. ⚠ Beachten Sie, dass mit der Scriptfunktion MODIFY_UC_OBJECT ausschließlich die früheste Startzeit verändert werden kann! Die Anzahl der Tage, nach denen die Aufgabe starten soll, wird bei Verwendung des Attributs EARLIEST_STARTTIME des Scriptsprachmittels immer auf 0 gesetzt. Ein bestimmter Wert kann nicht vergeben werden! Ab der Operations Manager Version 8 gibt es mit dem Scriptsprachmittel MODIFY_TASK die Möglichkeit den frühesten Startzeitpunkt einer Aufgabe in einem aktivierten Workflow mir dem Format "TT/HH:MM" einzustellen.
	REMOVE_DEPENDENCY Entfernung der Abhängigkeit zum direkten Vorgänger	Die Aufgabe ignoriert den erwarteten Endestatus des angegebenen Vorgängers. Das hat zur Folge, dass die Verarbeitung bei der Aufgabe fortgesetzt wird, sofern alle anderen Vorgänger schon beendet wurden. Syntax für den Parameter *Wert*: *Name der Aufgabe [, Name des Vorgängers]* Wenn Sie keinen bestimmten Vorgänger angeben, werden alle Vorgänger ignoriert und die Aufgabe startet sofort.
CALL, JOBF, JOBP, JOBS, JSCH, SCRI	PRIORITY Aufgabenpriorität	Erlaubte Werte: 0 bis 255
Workflow-Aufgaben	IGNORE_CONDITIONS	Startet eine Workflow-Aufgabe sofort, die sich im Status "Warten auf Pre-Conditions" befindet. Der Parameter Wert ist dabei nicht anzugeben.
Workflow-Aufgaben	CHECK_CONDITIONS	Prüfung der Bedingungen/Anweisungen einer Workflow-Aufgabe, die sich im Status "Warten auf Pre-Conditions" befindet, von vorne beginnen. Der Parameter Wert ist dabei nicht anzugeben.

CALL, EVNT, JOBF, JOBG, JOBP, JOBQ,JOBS, JSCH, SCRI	IGNORE_ QUEUE	Startet eine Aufgabe sofort, die sich auf Grund unzureichender Queue-Slots in einem Wartezustand befindet befindet. Der Parameter Wert ist dabei nicht anzugeben.
Alle aktivierbaren Objekte	IGNORE_SYNC	Die Sync-Einstellungen werden ignoriert. Der Wert spielt dabei keine Rolle, dennoch ist dieser Parameter anzugeben. Beispiel: `MODIFY_UC_OBJECT(&RUNNR#, IGNORE_SYNC, "")`
Periodische Aufgaben	GAP	Ändert bei einer periodischen Aufgabe den zeitlichen Abstand zwischen den Durchführungen. Wert in Minuten
Alle ausführbaren Objekte	SET_EXPRESS	Startet sofort eine Aufgabe, die sich im Zustand "Warten auf Ressource" befindet (JOBS, JOBF und JOBD). Erlaubte Werte: "ON" und "OFF"

Sie können eine Aufgabe nur dann mit dieser Script-Funktion ändern, wenn sie schon generiert ist, sich aber noch in einem Wartezustand befindet. Sie darf nicht die Option **Zur Laufzeit generieren** gesetzt haben. Verwenden Sie die Script-Anweisung :PUT_ATT, wenn eine Aufgabe bereits zum Generierungszeitpunkt verändert werden soll.

Mit der Script-Anweisung :ON_ERROR legen Sie die Reaktion auf Fehler fest. Wie bisher können Sie sie mit den Script-Funktionen zur Fehlerbehandlung analysieren. Das Script wird weiter prozessiert. Es besteht aber auch die Möglichkeit, die Prozessierung des Scripts abzubrechen.

Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

Beispiele

Das erste Beispiel zeigt, wie eine Benachrichtigung (Rückfrage) quittiert wird. Die laufende Nummer (RunID) muss dafür vorher ermittelt und in der Script-Variablen "&RUNNR#" gespeichert worden sein.

```
:SET &RUNNR# = ACTIVATE_UC_OBJECT("Nachtschicht")
!...
:SET &MODOBJ# = MODIFY_UC_OBJECT( &RUNNR#, RESPONSE, OK)
```

Im zweiten Beispiel wird eine Gruppe aktiviert und die laufende Nummer (RunID) in einer Script-Variablen gespeichert. Die Anzahl der parallel laufenden Aufgaben in der Gruppe wird auf "1" gesetzt. Vorgemerkte Aufgaben können danach nur seriell verarbeitet werden.

```
:SET &RUNNR# = ACTIVATE_UC_OBJECT("GRP7")
:SET &MODOBJ# = MODIFY_UC_OBJECT( &RUNNR#, GR_MAX_PAR_JOBS, 1)
```

Die folgende Beispiele zeigen Änderungen einer Aufgabe, die in einem Workflow läuft.

```
!Der Job "MAWI.ABSCHLUSS" soll sofort starten.
:SET &RUNNR# = GET_UC_OBJECT_NR(MAWI.TAG)
:SET &MODOBJ# = MODIFY_UC_OBJECT(&RUNNR#, GOIMM, "MAWI.ABSCHLUSS")
```

```
!Der Job "MAWI.ABSCHLUSS" soll frühestens erst um 15:00 laufen.
:SET &RUNNR# = GET_UC_OBJECT_NR(MAWI.TAG)
:SET &MODOBJ# = MODIFY_UC_OBJECT(&RUNNR#, EARLIEST_STARTTIME,
"MAWI.ABSCHLUSS","150000")

!Der Job "MAWI.ABSCHLUSS" soll die früheste Startzeit ignorieren.
:SET &RUNNR# = GET_UC_OBJECT_NR(MAWI.TAG)
:SET &MODOBJ# = MODIFY_UC_OBJECT(&RUNNR#, EARLIEST_STARTTIME,
"MAWI.ABSCHLUSS", OFF)

!Der Job "MAWI.ABSCHLUSS" soll die Abhängigkeit zum Vorgänger
"MAWI.TAGAUS" ignorieren.
:SET &RUNNR# = GET_UC_OBJECT_NR(MAWI.TAG)
:SET &MODOBJ# = MODIFY_UC_OBJECT(&RUNNR#, REMOVE_DEPENDENCY,
"MAWI.ABSCHLUSS", "MAWI.TAGAUS")
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:ADD_ATT	Fügt einem Benachrichtigungs-Objekt Empfänger zur Laufzeit hinzu.
:PUT_ATT	Ändert den Wert eines Attributes während der Generierung.
GET_ATT	Liefert den Wert von Attributen einer Aufgabe während der Generierung.
MODIFY_TASK	Ändert laufende AblaufPläne.
:ON_ERROR	Legt die Reaktion auf bestimmte Fehler bzw. Meldungen von Script-Sprachmitteln fest.

Script-Sprachmittel - Objekte lesen oder verändern
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.29 SET_SYNC

Script-Funktion: Führt die definierte Aktion eines Sync-Objektes aus.

Syntax

SET_SYNC*(Sync, Aktion)*

Syntaxteil	Beschreibung/Format
Sync	Name eines Sync-Objektes, dessen Aktion ausgeführt werden soll. Format: AE-name, Script-Literal oder Script-Variable
Aktion	Aktion eines Sync-Objektes, die ausgeführt werden soll. Format: AE-Name, Script-Literal oder Script-Variable

Rückgabewerte
"Y" - Die Aktion konnte beim Sync ausgeführt werden. "N" - Die Aktion konnte nicht beim Sync ausgeführt werden.

Anmerkungen

Die Aktionen zu einem Sync-Objekt können Sie in der Registerkarte "Sync" definieren.

Ob eine Aktion ausführbar ist oder nicht, hängt von diesen Definitionen ab, da jede Aktion nur bei einem bestimmten Zustand durchgeführt werden kann.

⚠ Beachten Sie, dass für die Durchführung des Sprachmittels Schreibrechte (W) auf das angegebene Sync-Objekt erforderlich sind!

Beispiel

Das folgende Beispiel setzt beim Sync-Objekt "DB.STATUS" die Aktion "EXKLUSIV".

```
:SET &RET#=SET_SYNC("DB.STATUS","EXKLUSIV")
:IF &RET# = "Y"
:   PRINT "Sync-Zustand EXKLUSIV wurde gesetzt!"
:ELSE
:   PRINT "Sync-Zustand EXKLUSIV wurde nicht gesetzt!"
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
GET_SYNC	Fragt den aktuellen Zustand oder Wert eines Sync-Objektes ab.
:ATTACH_SYNC	Weist einer Aufgabe ein Sync-Objekt zu.

Script-Sprachmittel - Objekte lesen oder verändern
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.30 XML_BEAUTIFY

Script-Funktion: Bereitet die Darstellung eines XML-Dokumentes auf.

Syntax

XML_BEAUTIFY *(Verweis)*

Syntaxteil	Beschreibung/Format
Verweis	Verweis auf ein XML-Dokument dessen auszugebende Struktur formatiert werden soll Format: Script-Literal oder Script-Variable

Rückgabewert
"0" - Der Inhalt des XML-Dokumentes wurde erfolgreich aufbereitet.

Anmerkungen

Die Script-Funktion bereitet die Darstellung eines XML-Dokumentes auf, bevor diese mit XML_PRINTINTOFILE in eine XML-Datei geschrieben wird.

Der Script-Funktion muss ein Verweis übergeben werden, um auf ein Element zugreifen zu können. Dieser wird von Script-Sprachmitteln bereitgestellt, die innerhalb des XML-Dokumentes positionieren können, beispielsweise XML_GET_FIRST_CHILD, XML_GET_NEXTSIBLING oder XML_SELECT_NODE. Die Script-Funktion bereitet dieses Element samt dessen Subelementen und die zugehörigen Attributen auf.

Es kann auch der Verweis verwendet werden, der XML_OPEN zurückgibt, um die Darstellung des gesamten XML-Dokumentes aufzubereiten.

Ohne die Verwendung von XML_BEAUTIFY werden die Informationen des XML-Dokumentes hintereinander (ohne Zeilenumbruch) in eine Datei geschrieben. Wird diese Datei mit einem Texteditor, beispielsweise Notepad, geöffnet, ist die Darstellung in Folge dessen unübersichtlich.

Beispiele

Die komplette Struktur der Dokumentation wird zunächst in besser lesbarer Form aufbereitet und anschließend in die Datei "Doku.xml" ausgegeben.

```
:SET&XMLDOCU# = XML_OPEN(DOCU,"BEREITSCHAFT","@Details")

:SET &RET1# = XML_BEAUTIFY(&XMLDOCU#)
:SET&RET2# = XML_PRINTINTOFILE("C:\AUTOMIC\XML_
Dokumentation\Doku.xml",&XMLDOCU#)

:XML_CLOSE
```

Das zweite Beispiel gibt die Elemente "Beschreibung" und "Objekte" mit ihren Attributen in eine Datei aus.

```
:SET&XMLDOCU# = XML_OPEN(DOCU,"BEREITSCHAFT","@Details")

:SET&HND# = XML_GET_FIRST_CHILD(&XMLDOCU#)
:SET &RET1# = XML_BEAUTIFY(&HND#)
:SET&RET2# = XML_PRINTINTOFILE("C:\AUTOMIC\XML_
Dokumentation\Doku.xml",&HND#)

:XML_CLOSE
```

Siehe auch:

XML_PRINTINTOFILE	Gibt das XML-Dokument in eine Datei aus

Script-Sprachmittel - Objekte lesen oder verändern
Strukturierte Dokumentation
www.w3c.org/TR/xmlbase
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.31 XML_GET_ATTRIBUTE

Script-Funktion: Liefert den Attributwert eines Elementes.

Syntax

XML_GET_ATTRIBUTE *(Verweis, @Attribut)*

Syntaxteil	Beschreibung/Format
Verweis	Verweis auf ein Element, dessen Attributwert bestimmt werden soll Format: Script-Literal oder Script-Variable
Attribut	Name des Attributes mit vorangestelltem "@" Format: Script-Literal oder Script-Variable

Rückgabewerte
Inhalt des Attributes. " " - Das Attribut besitzt keinen Wert bzw. das Element hat keine Attribute.

Anmerkungen

Mit dieser Script-Funktion kann der Attributwert eines Elementes in einem XML-Dokument gelesen werden.

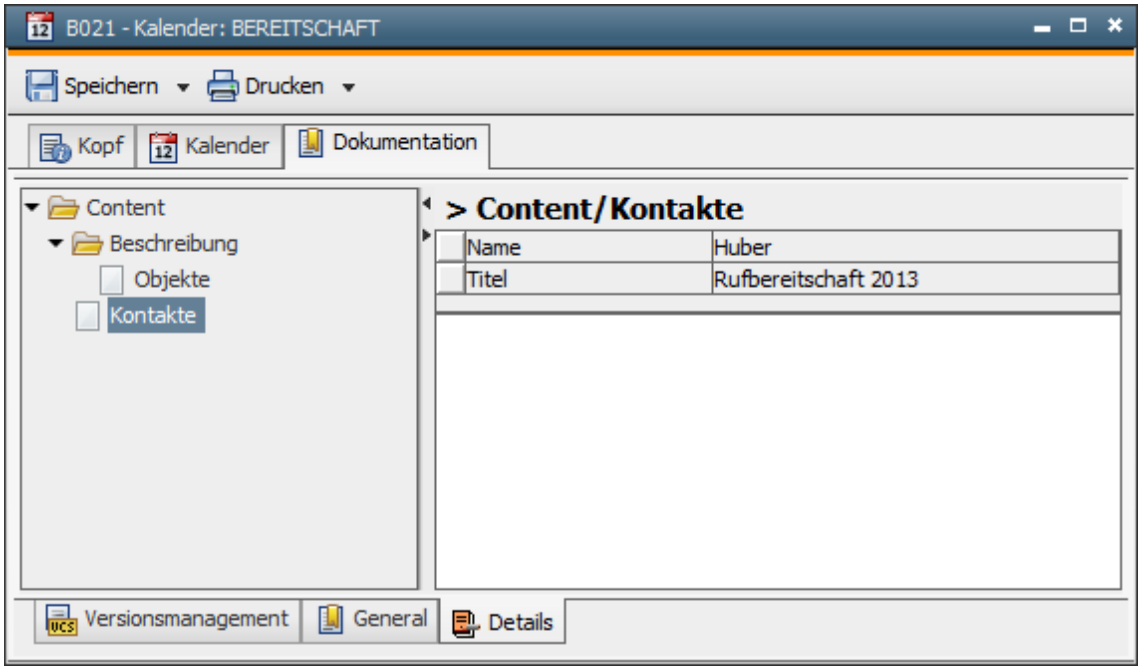
Der Script-Funktion muss ein Verweis übergeben werden, um auf das Element zugreifen zu können. Dieser wird von Script-Sprachmitteln bereitgestellt, die innerhalb eines XML-Dokumentes positionieren können, beispielsweise XML_GET_FIRST_CHILD, XML_GET_NEXTSIBLING, XML_SELECT_NODE oder XML_OPEN.

In der strukturierten Dokumentation wird zwischen den Attributarten "Text" und "Aufzählung" unterschieden, die gleichermaßen von XML_GET_ATTRIBUTE gelesen werden können.

Für den zurückgelieferten Wert gibt es keine Beschränkung.

Beispiel

Im folgenden Beispiel wird der Wert des Attributes "Name" (Typ Aufzählung) und der des Attributes "Titel" (Typ Text) ermittelt. Die Inhalte "Meier" und "Rufbereitschaft 2005" werden im Aktivierungsprotokoll ausgegeben.




```
:SET  &XMLDOCU# = XML_OPEN(DOCU,"BEREITSCHAFT","@Details")

:   SET  &HND# = XML_GET_FIRST_CHILD(&XMLDOCU#)
:   SET  &HND# = XML_GET_NEXTSIBLING(&HND#)
:   SET &NAME# = XML_GET_ATTRIBUTE(&HND#, "@Name")
:   PRINT "Attributwert: &NAME#"
:   SET &TITEL# = XML_GET_ATTRIBUTE(&HND#, "@Titel")
:   PRINT "Attributwert: &TITEL#"

:XML_CLOSE
```

Siehe auch:

Script-Sprachmittel - Objekte lesen oder verändern
Strukturierte Dokumentation
www.w3c.org/TR/xmlbase
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.32 XML_GET_CHILD_COUNT

Script-Funktion: Zählt Subelemente eines Elementes.

Syntax

XML_GET_CHILD_COUNT ***(Verweis)***

Syntaxteil	Beschreibung/Format

Verweis	Verweis auf ein Element, dessen Anzahl an Subelementen ermittelt werden soll Format: Script-Literal oder Script-Variable

Rückgabewert

Anzahl der Subelemente.

Anmerkungen

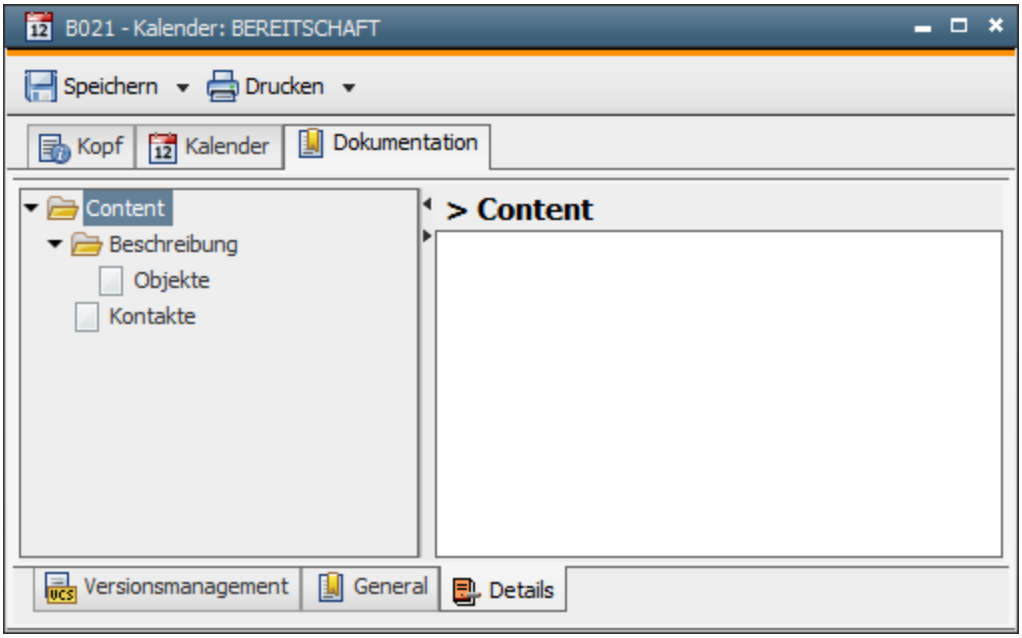
Die Script-Funktion ermittelt die Anzahl von Subelementen eines Elementes.

Der Script-Funktion muss ein Verweis übergeben werden, um auf das Element zugreifen zu können. Dieser wird von Script-Sprachmitteln bereitgestellt, die innerhalb des XML-Dokumentes positionieren können, beispielsweise XML_GET_FIRST_CHILD, XML_GET_NEXTSIBLING, XML_SELECT_NODE oder XML_OPEN.

⚠ Beachten Sie, dass nur die Subelemente gezählt werden, die sich in der nächsten Strukturtiefe befinden. Die Script-Funktion ermittelt nicht die Gesamtzahl der Subelemente eines Elementes.

Beispiel

Es wird die Anzahl der Subelemente "2" im Aktivierungsprotokoll ausgegeben.




```
:SET  &XMLDOCU# = XML_OPEN(DOCU,"BEREITSCHAFT","@Details")

:SET  &NR# = XML_GET_CHILD_COUNT(&XMLDOCU#)
:PRINT  "Anzahl der Subelemente von Content: &NR#"

:XML_CLOSE
```

Siehe auch:

Script-Sprachmittel - Objekte lesen oder verändern

Strukturierte Dokumentation
www.w3c.org/TR/xmlbase
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.33 XML_GET_FIRST_CHILD

Script-Funktion: Ermittelt das erste Subelement eines Elementes.

Syntax

XML_GET_FIRST_CHILD *(Verweis)*

Syntaxteil	Beschreibung/Format
Verweis	Verweis auf ein Element, dessen Subelement bestimmt werden soll Format: Script-Literal oder Script-Variable

Rückgabewert
Verweis auf das erste Subelement " " - Das Element besitzt kein Subelement.

Anmerkungen

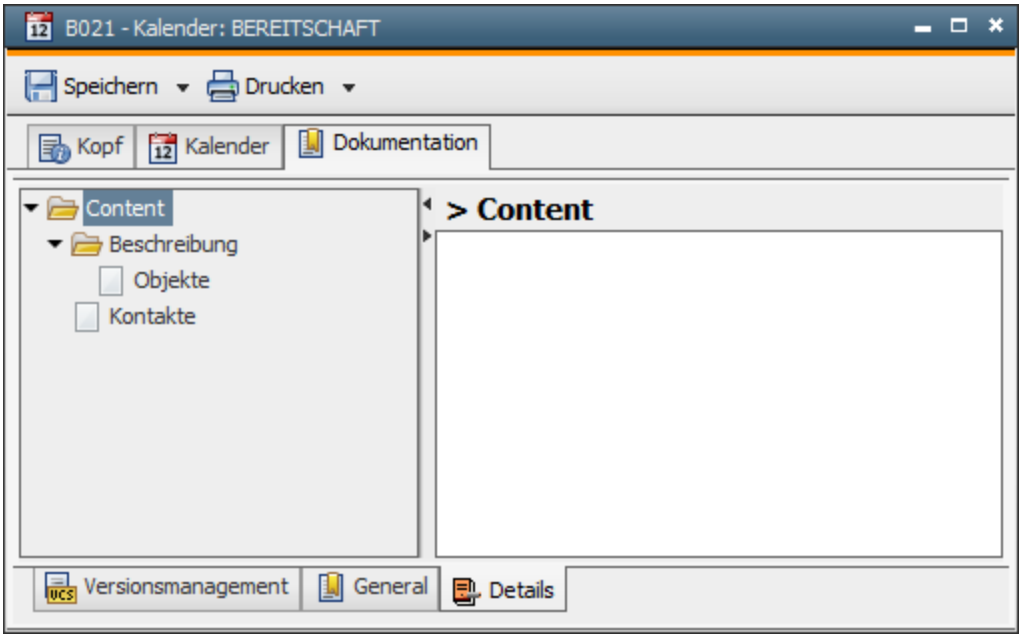
Die Script-Funktion ermittelt das jeweils erste Subelement eines Elementes in dem XML-Dokument.

Der Script-Funktion muss ein Verweis übergeben werden, um auf das jeweils erste Subelement zugreifen zu können. Dieser Verweis wird von speziellen Script-Sprachmitteln genutzt, um sich innerhalb des XML-Dokumentes zu positionieren und diese zu bearbeiten. Allererstes Handle ist jenes, das XML_OPEN zurückgibt. Es verweist beispielsweise auf das Root-Element "Content" der strukturierten Dokumentation.

Während mit XML_GET_FIRST_CHILD eine Ebene nach rechts in die Struktur hinein positioniert wird, kann XML_GET_NEXTSIBLING verwendet werden, um das auf gleicher Ebene nachfolgende Element zu ermitteln.

Beispiel

Im folgenden Beispiel wird der Name des ersten Subelementes "Beschreibung" bestimmt.




```
:SET  &XMLDOCU# = XML_OPEN(DOCU,"BEREITSCHAFT","@Details")

:SET  &HND# = XML_GET_FIRST_CHILD(&XMLDOCU#)

:   IF  &HND# <> ""
:      SET  &NAME# = XML_GET_NODE_NAME(&HND#)
:      PRINT  "Subelement: &NAME#"
:   ELSE
:      PRINT  "Kein Subelement vorhanden."
:   ENDIF

:XML_CLOSE
```

Siehe auch:

Script-Sprachmittel	Beschreibung
XML_GET_LAST_CHILD	Ermittelt das letzte Subelement eines Elementes.
XML_GET_NEXTSIBLING	Ermittelt das nachfolgende Element.

Script-Sprachmittel - Objekte lesen oder verändern
Strukturierte Dokumentation
www.w3c.org/TR/xmlbase
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.34 XML_GET_LAST_CHILD

Script-Funktion: Ermittelt das letzte Subelement eines Elementes.

Syntax

XML_GET_LAST_CHILD ***(Verweis)***

Syntaxteil	Beschreibung/Format
Verweis	Verweis auf ein Element, dessen Subelement bestimmt werden soll Format: Script-Literal oder Script-Variable

Rückgabewert
Verweis auf das letzte Subelement " " - Das Element besitzt kein Subelement.

Anmerkungen

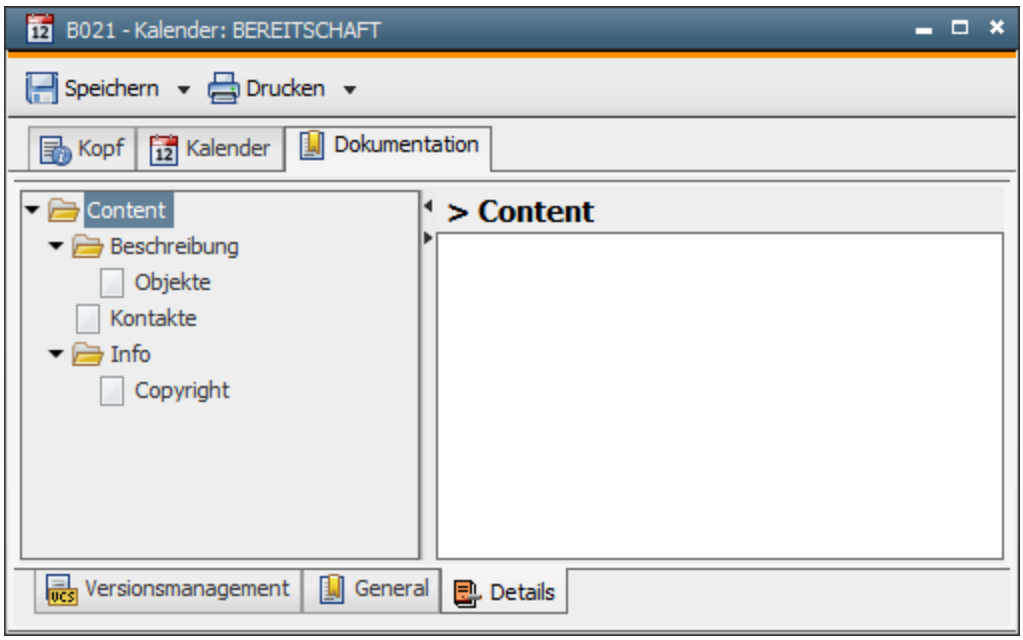
Die Script-Funktion ermittelt das letzte Subelement eines Elementes in dem XML-Dokument.

Der Script-Funktion muss ein Verweis übergeben werden, um auf das jeweils letzte Subelement zugreifen zu können. Dieser Verweis gibt die Position im XML-Dokument an. Allererstes Handle ist jenes, das XML_OPEN zurückgibt. Es verweist beispielsweise auf das Root-Element "Content" der strukturierten Dokumentation.

XML_GET_LAST_CHILD wechselt eine Stufe tiefer in die Struktur, ausgegangen vom angegebenen Element. XML_GET_NEXTSIBLING hingegen dient dazu, um auf gleicher Ebene das nachfolgende Element zu ermitteln.

Beispiel

Im folgenden Beispiel wird der Name des letzten Subelementes "Info" bestimmt, ausgegangen vom Hauptelement "Content".




```
:SET   &XMLDOCU# = XML_OPEN(DOCU,"BEREITSCHAFT","@Details")

:SET   &HND# = XML_GET_LAST_CHILD(&XMLDOCU#)

:    IF   &HND# <> ""
:       SET   &NAME# = XML_GET_NODE_NAME(&HND#)
:       PRINT   "Letztes Subelement: &NAME#"
```

```
:    ELSE
:       PRINT  "Kein Subelement vorhanden."
:    ENDIF

:XML_CLOSE
```

Siehe auch:

Script-Sprachmittel	Beschreibung
XML_GET_FIRST_CHILD	Ermittelt das erste Subelement eines Elementes.
XML_GET_NEXTSIBLING	Ermittelt das nachfolgende Element.

Script-Sprachmittel - Objekte lesen oder verändern
Strukturierte Dokumentation
www.w3c.org/TR/xmlbase
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.35 XML_GET_NEXTSIBLING

Script-Funktion: Ermittelt das nachfolgende Element.

Syntax

XML_GET_NEXTSIBLING *(Verweis)*

Syntaxteil	Beschreibung/Format
Verweis	Verweis auf Element, dessen Nachfolger ermittelt werden soll Format: Script-Literal oder Script-Variable

Rückgabewerte
Verweis auf das nachfolgende Element der gleichen Ebene. " " - Das Element besitzt kein nachfolgendes Element.

Anmerkungen

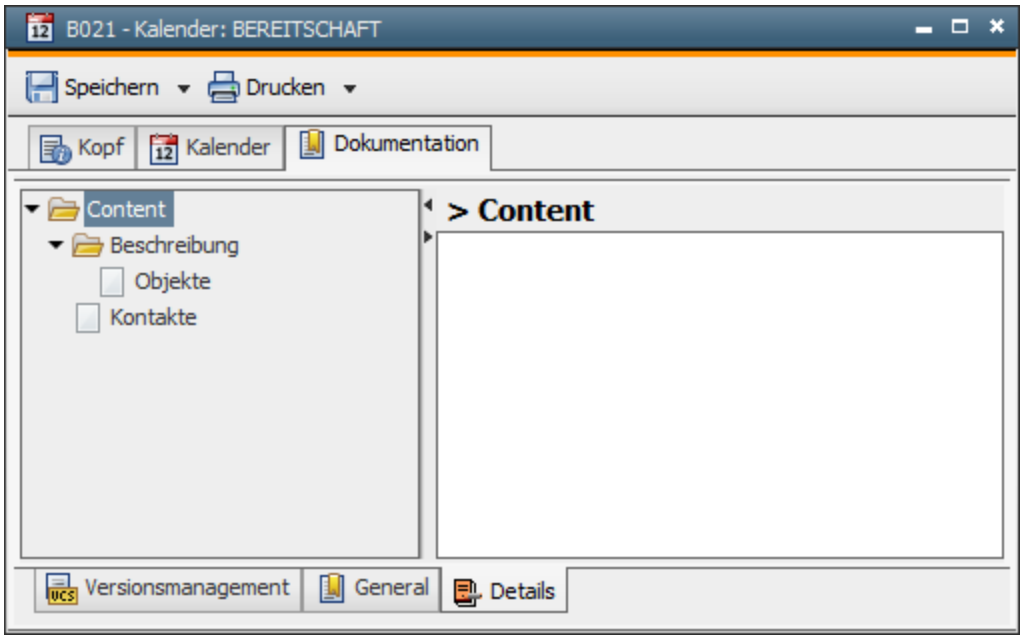
Die Script-Funktion ermittelt das auf gleicher Ebene nachfolgende Element in dem XML-Dokument.

Der Script-Funktion muss ein Verweis übergeben werden, um auf das nachfolgende Element zugreifen zu können. Dieser wird von speziellen Script-Sprachmitteln genutzt, um sich innerhalb des XML-Dokumentes zu positionieren und dieses zu bearbeiten.

Während XML_GET_NEXTSIBLING dazu dient, das nachfolgende Element auf der gleichen Ebene zu ermitteln, wird XML_GET_FIRST_CHILD verwendet, um das erste Subelement zu bestimmen.

Beispiel

Ausgangspunkt des Beispiels ist das erste Subelement namens "Beschreibung". Alle auf gleicher Ebene nachfolgenden Elemente werden ermittelt und im Aktivierungsprotokoll ausgegeben. In diesem Fall ist das nur das Element "Kontakte".




```
:SET   &XMLDOCU# = XML_OPEN(DOCU,"BEREITSCHAFT","@Details")

:SET   &HND# = XML_GET_FIRST_CHILD(&XMLDOCU#)

:    WHILE   &HND# <> ""
:       SET   &NAME# = XML_GET_NODE_NAME(&HND#)
:       PRINT   "Element: &NAME#"
:       SET &HND# = XML_GET_NEXTSIBLING(&HND#)
:    ENDWHILE

:XML_CLOSE
```

Siehe auch:

Script-Sprachmittel	Beschreibung
XML_GET_FIRST_CHILD	Ermittelt das erste Subelement eines Elementes

Script-Sprachmittel - Objekte lesen oder verändern
Strukturierte Dokumentation
www.w3c.org/TR/xmlbase
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.36 XML_GET_NODE_NAME

Script-Funktion: Liefert den Namen eines Elementes.

Syntax

XML_GET_NODE_NAME *(Verweis)*

Syntaxteil	Beschreibung/Format
Verweis	Verweis auf ein Element, dessen Name ermittelt werden soll Format: Script-Literal oder Script-Variable

Rückgabewert
Name des Elementes

Anmerkungen

Mit dieser Script-Funktion kann der Name eines Elementes in einem XML-Dokument gelesen werden.

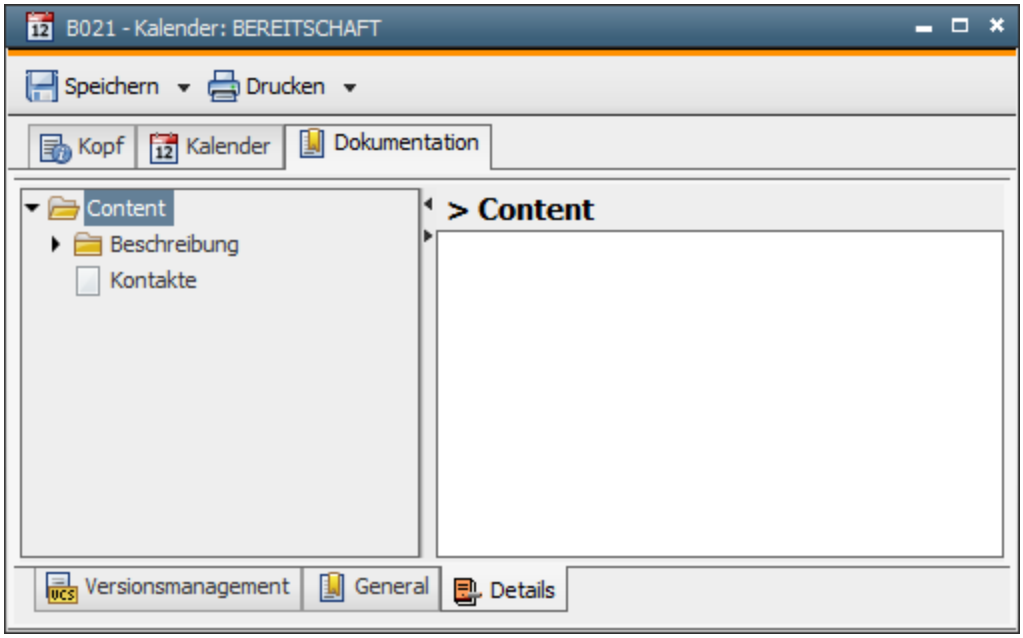
Der Script-Funktion muss eine Verweis auf das Element übergeben werden. Dieser wird von Script-Sprachmitteln bereitgestellt, die innerhalb des XML-Dokumentes positionieren können, beispielsweise XML_GET_FIRST_CHILD, XML_GET_NEXTSIBLING oder XML_SELECT_NODE.

XML_OPEN liefert einen Verweis auf das erste Element.

Bitte beachten Sie, dass die Script-Funktion nicht mehr als 1024 Zeichen verarbeiten kann und bei Überschreiten dieses Limits keinen Rückgabewert mehr liefert. Achten Sie deshalb auf die Länge des Elementnamens.

Beispiel

Das Beispiel ermittelt alle auf der gleichen Ebene liegende Elemente. Deren Namen "Beschreibung" und "Kontakte" werden im Aktivierungsprotokoll ausgegeben.




```
:SET  &XMLDOCU# = XML_OPEN(DOCU,"BEREITSCHAFT","@Details")

:SET  &HND# = XML_GET_FIRST_CHILD(&XMLDOCU#)

:   WHILE  &HND# <> ""
:      SET  &NAME# = XML_GET_NODE_NAME(&HND#)
:      PRINT  "Element: &NAME#"
:      SET &HND# = XML_GET_NEXTSIBLING(&HND#)
:   ENDWHILE

:XML_CLOSE
```

Siehe auch:

Script-Sprachmittel	Beschreibung
XML_GET_NODE_TEXT	Liefert den Text eines Elementes.

Script-Sprachmittel - Objekte lesen oder verändern
Strukturierte Dokumentation
www.w3c.org/TR/xmlbase
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.37 XML_GET_NODE_TEXT

Script-Funktion: Liefert den Text eines Elementes.

Syntax

XML_GET_NODE_TEXT ***(Verweis)***

Syntaxteil	Beschreibung/Format
Verweis	Verweis auf Element, dessen Text ermittelt werden soll Format: Script-Literal oder Script-Variable

Rückgabewerte
Text des Elementes. " " - Das Element besitzt keinen Text.

Anmerkungen

Mit dieser Script-Funktion kann der Text eines Elementes in einem XML-Dokument gelesen werden.

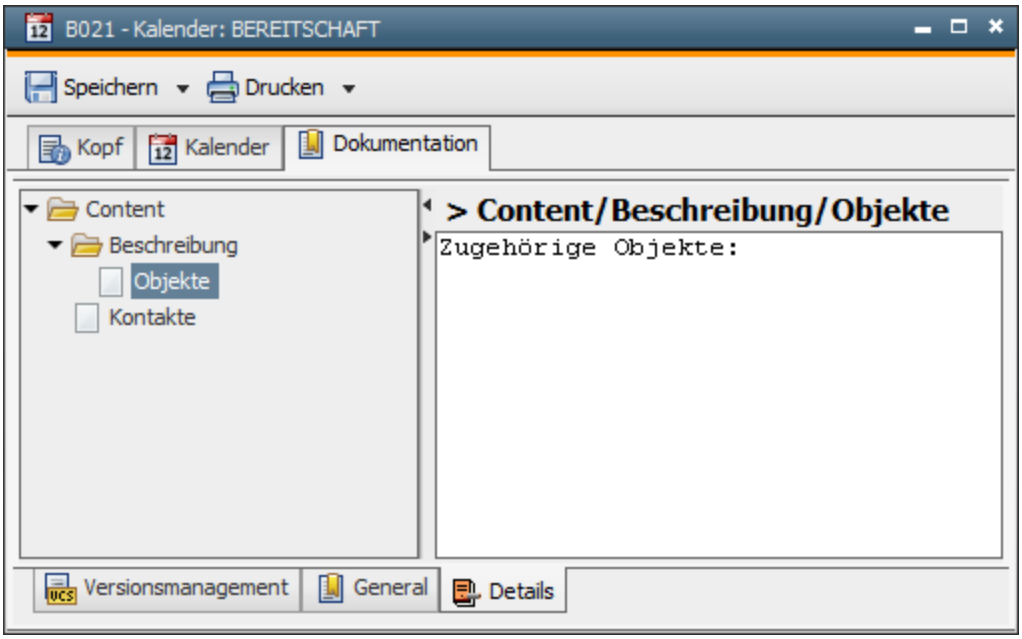
Der Script-Funktion muss ein Verweis übergeben werden, um auf das Element zugreifen zu können. Dieser wird von Script-Sprachmitteln bereitgestellt, die innerhalb des XML-Dokumentes positionieren können, beispielsweise XML_GET_FIRST_CHILD, XML_GET_NEXTSIBLING, XML_SELECT_NODE oder XML_OPEN.

Den Inhalt eines Text-Attributes erhalten Sie mit der Script-Funktion XML_GET_ATTRIBUTE.

Für den zurückgelieferten Wert gibt es keine Beschränkung.

Beispiel

Das Beispiel gibt den Text "Zugehörige Objekte:" des Elementes "Objekte" im Aktivierungsprotokoll aus.




```
:SET&XMLDOCU# = XML_OPEN(DOCU,"BEREITSCHAFT","@Details")

:SET&HND# = XML_GET_FIRST_CHILD(&XMLDOCU#)
:SET&HND# = XML_GET_FIRST_CHILD(&HND#)

:WHILE&HND# <> ""
:       SET&NAME# = XML_GET_NODE_NAME(&HND#)
```

```
:       SET&TXT# = XML_GET_NODE_TEXT(&HND#)
:       PRINT "Text von Element &NAME#: &TXT#"
:       SET&HND# = XML_GET_NEXTSIBLING(&HND#)
:ENDWHILE

:XML_CLOSE
```

Siehe auch:

Script-Sprachmittel	Beschreibung
XML_GET_NODE_NAME	Liefert den Namen eines Elementes.

Script-Sprachmittel - Objekte lesen oder verändern
Strukturierte Dokumentation
www.w3c.org/TR/xmlbase
Beispielsammlung
Ermittlung der Fehlermeldung und -nummer
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.38 XML_OPEN

Script-Funktion: Öffnet ein XML-Dokument für die Bearbeitung.

Syntax

XML_OPEN ***(*****DOCU*****,*** *[Objektname]****, @Dokumentation)***

XML_OPEN ***(*****REPORT*****,*** *[RunID]****, Report)***

XML_OPEN ***(*****STRING*****, Zeichenkette)***

Syntaxteil	Beschreibung/Format
Quelle	Herkunft des XML-Dokumentes Format: AE-name, Script-Literal oder Script-Variable Erlaubte Werte: "DOCU", "REPORT" und "STRING" "DOCU" - Strukturierte Dokumentation "REPORT" - Report eines SAP-Jobs "STRING" - XML in Form einer Zeichenkette
Für strukturierte Dokumentationen	
Objektname	Name des Objektes Format: Script-Literal oder Script-Variable Handelt es sich um die Registerkarte des eigenen Objektes, so ist dieser Parameter optional.
Dokumentation	Name der Dokumentations-Registerkarte mit vorangestelltem "@" Format: Script-Literal oder Script-Variable
Für Reports	

RunID	Laufende Nummer (RunID) der Aufgabe, deren Report geöffnet werden soll. Format: Script-Literal oder Script-Variable Handelt es sich um den Report des eigenen Objektes, so ist dieser Parameter optional.
Report	Typ des XML-Reports
Für Zeichenketten	
Zeichenkette	Zeichenkette, welche das XML enthält. Format: Script-Variable oder Script-Literal

Rückgabewert
Verweis auf das XML-Dokument

Anmerkungen

Die Script-Funktion öffnet ein XML-Dokument für die Bearbeitung. Dabei kann es sich um eine strukturierte Dokumentation , um den Report eines SAP-Jobs handeln. Auch möglich ist die direkte Angabe von XML in Form einer Script-Variable oder Script-Literals.

Um Objekten nähere Beschreibungen und Erklärungen hinzuzufügen, können Registerkarten zur Dokumentation verwendet werden. Diese sind in der Variablen UC_OBJECT_DOCU für die einzelnen Objekttypen definiert. Eine spezielle Form der Darstellung ist die strukturierte Dokumentation, welche durch ein vorangestelltes "@" gekennzeichnet wird. Die strukturierte Dokumentation können Sie mit speziellen Script-Sprachmitteln auslesen.

Reporte von SAP-Jobs werden als XML-Dokumente abgelegt. Sie enthalten beispielsweise für XI_GET_CHANNEL eine Liste der selektierten Kommunikationskanäle.

Der Rückgabewert der Script-Funktion ist ein Handle, ein Verweis auf das XML-Dokument. Dieser Handle wird von einigen Script-Sprachmitteln, mit denen Sie das XML-Dokument bearbeiten, als Parameter verwendet.

:XML_CLOSE schließt das XML-Dokument.

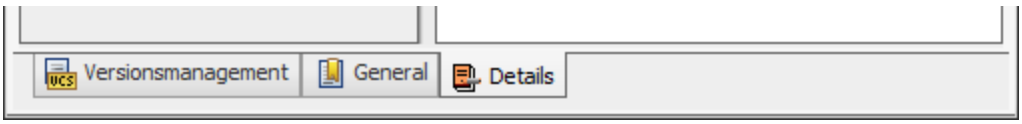
Es kann immer nur ein XML-Dokument zur gleichen Zeit geöffnet werden.

Diese Script-Funktion wurde in einer früheren Version der Automation Enginevon XML_OPEN_DOCU auf XML_OPEN umbenannt. Die alte Schreibweise jedoch wird weiterhin unterstützt.

Beispiel

Die Registerkarte "Details" des Objektes "MAWI.TAG" wird zur weiteren Bearbeitung geöffnet.

```
:SET &XMLDOCU# = XML_OPEN(DOCU,"MAWI.TAG", "@Details")
```



Siehe auch:

Script-Sprachmittel	Beschreibung
:XML_CLOSE	Schließt ein XML-Dokument.

Script-Sprachmittel - Objekte lesen oder verändern
Strukturierte Dokumentation
www.w3c.org/TR/xmlbase
Beispielsammlung
Ermittlung der Fehlermeldung und -nummer
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.39 XML_PRINTINTOFILE

Script-Funktion: Gibt Struktur von Elementen in eine XML-Datei aus.

Syntax

XML_PRINTINTOFILE *(Datei, Verweis)*

Syntaxteil	Beschreibung/Format
Datei	Name und Pfad der XML-Datei Format: Script-Literal oder Script-Variable
Verweis	Verweis auf Element, dessen Struktur ausgegeben werden soll Format: Script-Literal oder Script-Variable

Rückgabewert
"0" - Die Struktur wurde erfolgreich in die Datei geschrieben.

Anmerkungen

Die Script-Funktion schreibt die Struktur eines Elementes und aller Subelemente in eine angegebene XML-Datei.

Der Script-Funktion muss ein Verweis übergeben werden, um auf ein Element zugreifen zu können. Dieser wird von Script-Sprachmitteln bereitgestellt, die innerhalb des XML-Dokumentes positionieren können, beispielsweise XML_GET_FIRST_CHILD, XML_GET_NEXTSIBLING oder XML_SELECT_NODE. Die Script-Funktion gibt dieses Element samt dessen Subelementen und die zugehörigen Attributen in eine Datei aus.

Es kann auch der Verweis verwendet werden, der XML_OPEN zurückgibt, um die gesamte Struktur auszugeben.

Alle Informationen zur Struktur des Elementes werden hintereinander (ohne Zeilenumbruch) in die angegebene Datei geschrieben. Wird diese Datei mit einem Texteditor, beispielsweise Notepad, geöffnet, ist die Darstellung in Folge dessen unübersichtlich. XML_BEAUTIFY kann vor der Script-Funktion verwendet werden, um den Inhalt der Datei zu formatieren und somit besser lesbar

darzustellen. Die Datei kann aber auch beispielsweise mit dem Microsoft Internet Explorer angezeigt werden. Dabei wird die Struktur des Elementes anschaulich dargestellt.

⚠ Beachten Sie, dass eine bereits existierende Datei ohne Nachfrage überschrieben wird.

Beispiel

Die komplette Struktur der Dokumentation wird zunächst in besser lesbarer Form aufbereitet und anschließend in die Datei "Doku.xml" ausgegeben.

```
:SET&XMLDOCU# = XML_OPEN(DOCU,,"@Details")

:SET &RET1# = XML_BEAUTIFY(&XMLDOCU#)
:SET&RET2# = XML_PRINTINTOFILE("C:\AUTOMIC\XML_
Dokumentation\Doku.xml",&XMLDOCU#)

:XML_CLOSE
```

Das zweite Beispiel gibt die Elemente "Beschreibung" und "Objekte" mit ihren Attributen in eine Datei aus.

```
:SET&XMLDOCU# = XML_OPEN(DOCU,"BEREITSCHAFT","@Details")

:SET&HND# = XML_GET_FIRST_CHILD(&XMLDOCU#)
:SET &RET1# = XML_BEAUTIFY(&HND#)
:SET&RET2# = XML_PRINTINTOFILE("C:\AUTOMIC\XML_
Dokumentation\Doku.xml",&HND#)

:XML_CLOSE
```

Siehe auch:

Script-Sprachmittel	Beschreibung
XML_BEAUTIFY	Bereitet die Darstellung einer Elementstruktur auf.

Script-Sprachmittel - Objekte lesen oder verändern
Strukturierte Dokumentation
www.w3c.org/TR/xmlbase
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.4.40 XML_SELECT_NODE

Script-Funktion: Ermittelt beliebiges Element.

Syntax

XML_SELECT_NODE*(Verweis, Element)*

Syntaxteil	Beschreibung/Format

Verweis	Verweis auf das Element, das als Ausgangspunkt dient Format: Script-Literal oder Script-Variable
Element	Name des gesuchten Elementes mit Pfad ab Ausgangspunkt Format: Script-Literal oder Script-Variable

Rückgabewerte
Verweis auf das gesuchte Element. " " - Das gesuchte Element existiert nicht.

Anmerkungen

Die Script-Funktion ermittelt ein beliebiges Element in dem XML-Dokument.

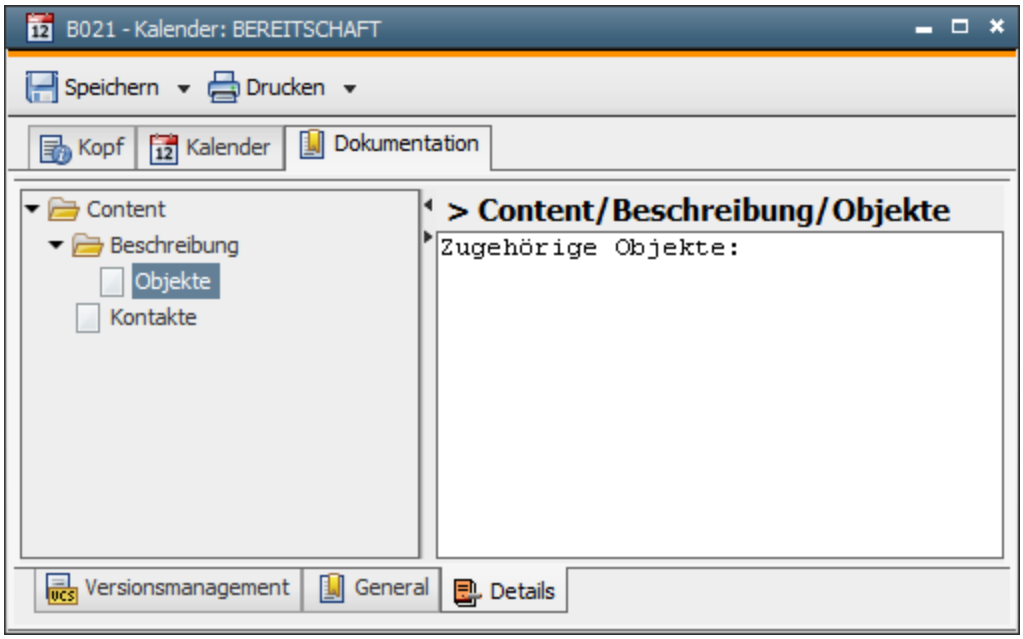
Der Script-Funktion muss ein Verweis übergeben werden, um zunächst auf ein Element zu positionieren. Dieses Element ist der Ausgangspunkt der Suche in der Struktur. Von dieser Stelle aus wird das Element gesucht, welches mit Pfad angegeben wurde.

Verweise werden von speziellen Script-Sprachmitteln genutzt, um sich innerhalb des XML-Dokumentes zu positionieren und diese zu bearbeiten. Der allererste Verweis ist jener, der XML_OPEN zurückgibt. Es verweist auf das Root-Element. Es kann von XML_SELECT_NODE verwendet werden, um direkt von dort aus ein Element zu ermitteln. Die Script-Funktion kann den Verweis allerdings auch von XML_GET_FIRST_CHILD oder XML_GET_NEXTSIBLING erhalten. Daraus ergeben sich flexible Möglichkeiten, um auf Elemente in stark verzweigten Strukturen zuzugreifen.

⚠ Beachten Sie, dass nur Elemente gefunden werden, die ein Subelement des Ausgangspunktes sind.

Beispiel

Das Beispiel ermittelt ein Element direkt ab dem Root-Element "Content". Der Text des Elementes "Objekte" wird im Aktivierungsprotokoll ausgegeben.



```
:SET  &XMLDOCU# = XML_OPEN(DOCU,"BEREITSCHAFT","@Details")

:SET &HND# = XML_SELECT_NODE(&XMLDOCU#, "Beschreibung/Objekte")
:SET  &TEXT# = XML_GET_NODE_TEXT(&HND#)
:PRINT  "Text des Elementes: &TEXT#"

:XML_CLOSE
```

Siehe auch:

Script-Sprachmittel - Objekte lesen oder verändern
Strukturierte Dokumentation
www.w3c.org/TR/xmlbase
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5 Script-Aufbau und -Prozessierung

3.5.1 :CLEAR

Script-Anweisung: Setzt ein Script-Array wieder auf seine Initialwerte zurück.

Syntax

:CLEAR *Script-Array*

Syntaxteil	Beschreibung/Format
Script-Array	Variablenname des Script-Arrays, welches zurückgesetzt werden soll. Format: Script-Variable

Anmerkungen

Durch diese Script-Anweisung kann ein bereits verwendetes Script-Array auf seine Initialwerte zurückgesetzt werden. Dadurch werden alle Werte gelöscht und das Array auf den selben Zustand zurückgesetzt, den es direkt nach Deklaration (:DEFINE) hatte.

Das Array ist dabei inklusive der leeren Indexklammern [] anzugeben.

Das Sprachmittel löscht nur den Inhalt des Arrays, die Variable selbst bleibt jedoch bestehen. Eine anschließende neue Definition des Arrays ist daher nicht möglich.

Beispiel

Folgendes Beispiel definiert ein Array und befüllt es mit den Werten einer Zeile eines statischen Variablen-Objektes. Diese Werte werden im Aktivierungsreport ausgegeben. Anschließend wird

das Array geleert und mit einem anderen Variablen-Eintrag befüllt. Die neuen Werte werden anschließend ebenfalls in das Aktivierungsprotokoll geschrieben.

```
:DEFINE &ARRAY#, string, 5
:DEFINE &STR#, string
:FILL &ARRAY#[] = GET_VAR(VARA.STATIC.TEST,"KEY01")
:SET &STR# = ARRAY_2_STRING(&ARRAY#[])
:P "KEY01: &STR#"
:CLEAR &ARRAY#[]
:FILL &ARRAY#[] = GET_VAR(VARA.STATIC.TEST,"KEY02")
:SET &STR# = ARRAY_2_STRING(&ARRAY#[])
:P "KEY02: &STR#"
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:DEFINE	Deklaration einer Script-Variable mit einem bestimmten Datentyp.

Script-Sprachmittel - Script-Aufbau und -Prozessierung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5.2 :CONST, :CONSTANT

Script-Anweisung: Erstellt eine Script-Variable als Konstante mit einem bestimmten Wert.

Syntax

:C[ONST[ANT]] ***Variablenname = Wert***

Syntaxteil	Beschreibung/Format
Variablenname	Name der Script-Variable, die als Konstante definiert werden soll. Der Name einer Script-Variablen besteht aus maximal 32 alphanumerischen Zeichen einschließlich der Sonderzeichen "$", "_", "@", "§" und "#". Umlaute sind nicht erlaubt. Das erste Zeichen des Namens darf keine Zahl sein. Im Script sind Variablen immer mit dem Kennzeichen "&" vor dem Namen anzugeben! Format: Script-Variable
Wert	Wert der Konstanten. Format: Script-Literal, Script-Variable oder Script-Funktion Dieser Wert ist nachträglich nicht änderbar!

Anmerkungen

Das Sprachmittel erstellt eine neue Variable als Konstante und weist dieser einen bestimmten Wert zu. Im Gegensatz zu Variablen, die mit :SET angelegt werden, ist der Wert von Konstanten schreibgeschützt und nachträglich nicht änderbar.

⚠ Beachten Sie, dass dieses Sprachmittel keine bestehende Script-Variable in eine Konstante umwandeln kann! Der Name von existierenden Variablen kann daher nicht angegeben werden.

⚠ Konstanten können nicht zuvor mit :DEFINE erstellt werden und besitzen somit keinen bestimmten Datentyp!

⚠ Bitte beachten Sie, dass `:CONST` Gleitkommazahlen automatisch auf- oder abrundet. Um Gleitkommazahlen zu speichern, nutzen Sie :DEFINE.

Beispiele

```
:CONST &COMPANY# = "Automic software"
:CONST &PI# = 3.1415
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:DEFINE	Deklariert eine Script-Variable mit einem bestimmten Datentyp.
:PSET	Weist einer Objektvariablen einen Wert zu.
:SET	Weist einer Script-Variablen einen Wert zu.
:SET_SCRIPT_VAR	Setzt die Werte von Script-Variablen per indirektem Zugriff.

Script-Sprachmittel - Script-Aufbau und -Prozessierung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5.3 :DATA

Script-Anweisung: Deklariert eine Scriptzeile explizit als DATA-Zeile.

Syntax

:DATA *Anweisung*

Syntaxteil	Beschreibung/Format
Anweisung	Anweisung für das jeweilige Zielsystem. Format: Script-Literal oder Script-Variable

Anmerkungen

DATA-Zeilen enthalten Anweisungen für Verarbeitungsschritte im jeweiligen Zielsystem.

Alle Zeilen des Scripts, welche keine Kommentarzeilen (beginnen mit einem Ausrufezeichen) oder Script-Anweisungen (werden mit einem Doppelpunkt eingeleitet) sind, werden automatisch als DATA-Zeilen betrachtet. Beginnt nun eine solche DATA-Zeile selbst mit einem Doppelpunkt, kommt es zu Konflikten, weil auch die Script-Anweisungen durch einen Doppelpunkt gekennzeichnet sind. In diesem Fall ist die DATA-Zeile explizit mit der :DATA-Anweisung zu deklarieren.

Es ist möglich, die *Anweisung* auch ohne Hochkommata zu schreiben. In diesem Fall werden alle Zeichen bis zum Zeilenende berücksichtigt. Die in *Anweisung* enthaltene Script-Variablen werden ersetzt.

⚠ DATA-Zeilen können nur im Objekttyp "Job" angewendet werden.

Beispiel

Dieses Beispiel aus einer PC-Batchverarbeitung definiert eine unbedingte Verzweigung zu einem Label, ab dem eine Fehlerbehandlung durchzuführen ist. Da Labels in einer PC-Batchverarbeitung mit einem Doppelpunkt beginnen, muss diese DATA-Zeile explizit mit der :DATA-Anweisung eingeleitet werden.

```
!...
GOTO FEHLERMELDUNG
!...
:DATA ":FEHLERMELDUNG"
!...
```

Siehe auch:

Script-Sprachmittel - Script-Aufbau und -Prozessierung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5.4 :DEFINE

Script-Anweisung: Deklaration einer Script-Variable mit einem bestimmten Datentyp.

Syntax

:D[EFINE] ***Variablenname, Datentyp*** *[, Array-Größe]*

Syntaxteil	Beschreibung/Format

Variablenname	Name der Script-Variable, die mit Angabe eines Datentyps neu erstellt wird. Der Name einer Script-Variablen besteht aus maximal 32 alphanumerischen Zeichen einschließlich der Sonderzeichen "$", "_", "@", "§" und "#". Umlaute sind nicht erlaubt. Das erste Zeichen des Namens darf keine Zahl sein. Im Script sind Variablen immer mit dem Kennzeichen "&" vor dem Namen anzugeben! Format: Script-Variable
Datentyp	Datentyp der deklarierten Variable. Erlaubte Werte: "unsigned", "string", "float" oder "signed" unsigned: Positive Ganzzahlen ohne Vorzeichen signed: Ganzzahlen mit Vorzeichen float: Gleitkommazahlen string: Zeichenkette, Text Die Angabe von Datentypen erfolgt nicht mit Hochkomma!
Array-Größe	Wenn die Script-Variable als Array erzeugt werden soll, wird mit diesem Parameter die Größe festgelegt. Format: Zahl ohne Hochkommata Erlaubte Werte: 1 bis 99999

Anmerkungen

Es ist möglich, Script-Variablen direkt mit dem Scriptsprachmittel :SET zu erstellen. Dabei erfolgt auch gleichzeitig die Wertzuweisung. Dies ist jedoch nur zulässig, wenn die angegebenen Werte den Datentypen "string" oder "unsigned" entsprechen! Wird der Variable eine Gleitkommazahl oder eine negative Zahl zugewiesen, kommt es zu einem Scriptfehler.

⚠ Eine Deklaration von Variablen, die bereits verwendet wurden, ist nicht möglich! Dies gilt für Script- und Objekt-Variablen, inklusive Definitionen in der Registerkarte "Variablen & Prompts"!

⚠ Die direkte Wertzuweisung eines anderen Datentyps funktioniert nur bei Script-Variablen, die mit :SET erstellt wurden und nur mit positiven Ganzzahlen und Strings!

⚠ Wird einer Script-Variable ein Wert zugewiesen, der außerhalb des erlaubten Bereichs des definierten Datentyps liegt, wird der Wert automatisch in den Zieltyp konvertiert.

⚠ Wird bei der Verwendung des Scriptsprachmittels :DEFINE kein oder ein ungültiger Datentyp angegeben, kommt es zu einem Scriptfehler.

Die verschiedenen Datentypen besitzen folgende Wertebereiche:

Datentyp	Wertebereich
unsigned	0 bis +9 999 999 999 999 999
signed	-9 999 999 999 999 999 bis +9 999 999 999 999 999
string	Alphanumerische Zeichenkette mit beliebiger Länge.
float	-9 999 999 999 999 999.9999999999999999 bis +9 999 999 999 999 999.9999999999999999

Bei Angabe des Parameters *Array-Größe* wird die Script-Variable als Array mit der festgelegten Länge erzeugt. Die Größe darf dabei den Wert 99999 nicht überschreiten. Beachten Sie die Besonderheiten im Umgang mit Script-Arrays. Die Indexklammern [] werden bei der Array-Deklaration nicht benötigt.

Verwenden Sie die Script-Anweisung :FILL, um Script-Arrays mit mehreren verschiedenen Werten zu befüllen.

Beispiele

Deklaration von mehreren Variablen mit verschiedenen Datentypen und anschließender Wertzuweisung:

```
:DEFINE  &a#, unsigned
:DEFINE&b#, signed
:DEFINE&c#, float
:DEFINE&d#, string
:SET&a# = 12
:SET&b# = -5
:SET&c# = 0.50
:SET&d# = "string"
```

In folgendem Beispiel wird der Datentyp des Variablenwertes automatisch in den Zieltyp konvertiert:

```
:DEFINE  &unsigned#, unsigned
:DEFINE  &string#, signed
:SET  &unsigned# = 12
:SET  &string# = &unsigned#
```

In diesem Beispiel ist die Zuweisung gültig, da die Variable "&unsigned#" mit der Script-Funktion CONVERT in den richtigen Datentyp umgewandelt wird:

```
:DEFINE&unsigned#, unsigned
:DEFINE&string#, signed
:SET&unsigned# = 12
:SET&string# = CONVERT(string,&unsigned#)
```

Wird eine Script-Variable mit dem Scriptsprachmittel :SET erstellt, ist eine direkte Wertzuweisung der Datentypen "unsigned" und "string" möglich.

```
:SET&setvar#= 12
:SET&setvar2#="string"
:SET&setvar# = &setvar2#
```

Dieses Beispiel zeigt die Erstellung und Befüllung eines Arrays mit Werten aus einem Variablen-Objekt.

```
:DEFINE&array#, unsigned, 5
:FILL&array#[] = GET_VAR(VARA1, ARRAY)
```

Folgendes Beispiel rundet das Ergebnis der Rechenoperation 3.0 - 0.1 in einen String unter Berücksichtigung einer Nachkommastelle mit Hilfe der Script-Funktion FORMAT und eines allgemeinen Rechenfehlers Epsilon = 0.0000005.

```
:DEFINE &SCRIPT_VERSION_ORIG#, float
:DEFINE &SCRIPT_VERSION_ORIG1#, float
:DEFINE &EPSILON#, float

:SET &SCRIPT_VERSION_ORIG# = 3.0
:SET &EPSILON# = 0.0000005
```

```
:P &SCRIPT_VERSION_ORIG#
:SET &SCRIPT_VERSION_ORIG# = &SCRIPT_VERSION_ORIG# - 0.1
:P BEFORE FORMAT: &SCRIPT_VERSION_ORIG#
:SET &SCRIPT_VERSION_ORIG# = &SCRIPT_VERSION_ORIG# + &EPSILON#
:SET &SCRIPT_VERSION_ORIG# = FORMAT(&SCRIPT_VERSION_ORIG#, "0.0")
:P AFTER FORMAT: &SCRIPT_VERSION_ORIG#
```

Sie finden detaillierte Ausführungen zum Problem der Darstellung von Gleitkommazahlen und deren Format im Kapitel "Gleitkommazahlen in der Automation Engine".

Siehe auch:

Script-Sprachmittel	Beschreibung
:SET	Weist einer Script-Variablen einen Wert zu.

Script-Sprachmittel - Script-Aufbau und -Prozessierung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5.5 :EXT_REPORT_OFF

Script-Anweisung: Schaltet das Protokollieren des Scripts einer Aufgabe aus.

Syntax

:EXT_REPORT_OFF

Anmerkung

Unter die erweiterten Reports fällt auch der Script-Report, der sämtliche Scriptzeilen des Objektes enthält. Standardmäßig wird er nicht angelegt und ist daher vom Administrator in der Variablen UC_CLIENT_SETTINGS mit dem Key "EXT_REPORTS" einzuschalten. Es ist auch möglich dies bei einem einzelnen Objekt in der Registerkarte **Kopf** zu tun.

Verwenden Sie die Script-Anweisung :EXT_REPORT_OFF, wenn einzelne Scriptzeilen oder das gesamte Script nicht in den Report aufgenommen werden sollen. Das Protokollieren wird bis zum Scriptende oder der Anweisung :EXT_REPORT_ON unterdrückt.

Im Script-Report wird eine Hinweiszeile ausgegeben, die anzeigt wieviele Scriptzeilen nicht protokolliert wurden.

```
**** EXT_REPORT_OFF_ON: 0005 lines suppressed ***
```

Beachten Sie, dass auch die Inhalte von Includes protokolliert werden, sofern bei deren Aufruf nicht der Parameter EXT_REPORT=OFF verwendet wurde.

Beispiel

Im Beispiel wird das Protokollieren für einen Teil des Scripts unterdrückt.

```
:EXT_REPORT_OFF
:INC JOCO2UC.SET.GLOBALS
:PUT_ATT FT_SRC_FILE = '$&JOCO_USERID.&COB85_ERRLST_PREF_
BS2.&PRGMNAME'
:PUT_ATT FT_DST_FILE = '&PC_PATH\&PRGMNAME_ERRLST.TXT'

:EXT_REPORT_ON
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:EXT_REPORT_ON	Schaltet das Protokollieren des Scripts einer Aufgabe ein.

Script-Sprachmittel - Script-Aufbau und -Prozessierung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5.6 :EXT_REPORT_ON

Script-Anweisung: Schaltet das Protokollieren des Scripts einer Aufgabe ein.

Syntax

:EXT_REPORT_ON

Anmerkung

Unter die erweiterten Reports fällt auch der Script-Report, der sämtliche Scriptzeilen des Objektes enthält. Standardmäßig wird er nicht angelegt und ist daher vom Administrator in der Variablen UC_CLIENT_SETTINGS mit dem Key "EXT_REPORTS" einzuschalten. Es ist auch möglich dies bei einem einzelnen Objekt in der Registerkarte **Kopf** zu tun.

Verwenden Sie die Script-Anweisung :EXT_REPORT_OFF, wenn einzelne Scriptzeilen oder das gesamte Script nicht in den Report aufgenommen werden sollen. Das Protokollieren wird bis zum Scriptende oder der Anweisung :EXT_REPORT_ON unterdrückt.

Beachten Sie, dass auch die Inhalte von Includes protokolliert werden, sofern bei deren Aufruf nicht der Parameter EXT_REPORT=OFF verwendet wurde.

Beispiel

Im Beispiel wird das Protokollieren für einen Teil des Scripts unterdrückt.

```
:EXT_REPORT_OFF
:INC JOCO2UC.SET.GLOBALS
:PUT_ATT FT_SRC_FILE = "$&JOCO_USERID.&COB85_ERRLST_PREF_
BS2.&PRGMNAME"
:PUT_ATT FT_DST_FILE = "&PC_PATH\&PRGMNAME_ERRLST.TXT"

:EXT_REPORT_ON
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:EXT_REPORT_OFF	Schaltet das Protokollieren des Scripts einer Aufgabe aus.

Script-Sprachmittel - Script-Aufbau und -Prozessierung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5.7 :FILL

Script-Anweisung: Speichert mehrere Werte in ein Script-Array.

Syntax

:FILL Script-Array =Werte

Syntaxteil	Beschreibung/Format
Script-Array	Name der Script-Variable, die als Array definiert wurde. Die Indexklammern sind dabei ohne Inhalt anzugeben. Format: Script-Variable
Werte	Werte die in das Array gespeichert werden sollen. Dazu sind die Script-Funktionen GET_PROCESS_LINE oder GET_VAR zu verwenden. Format: Script-Funktion

Anmerkungen

Die Script-Funktion dient dazu, um mehrere Werte gleichzeitig in ein Script-Array abzulegen. Arrays müssen vor der Wertezuweisung mit :DEFINE unter Angabe der Array-Größe erstellt werden. Für die Übergabe der Werte sind die Script-Funktionen GET_PROCESS_LINE, STR_SPLIT oder GET_VAR zu verwenden. Die Werte werden dabei immer ab dem ersten Elemente (Index = 1) in das Array geschrieben.

Übersteigt die Anzahl der zu speichernden Werte die Kapazität des Arrays, werden nur die ersten Werte abgelegt, die Platz im Array finden. Ist das Array größer als die ermittelten Werte, bleiben die restlichen Werte unverändert.

Das Script-Array ist für die Funktion mit leeren Indexklammern [] anzugeben.

Verwenden Sie :PUBLISH, um Arrays an unter- und übergeordnete Objekte weiter zu geben.

Beispiel

Im ersten Beipiel werden Werte eines Variablen-Objektes mit GET_VAR in das Array gespeichert.

```
:DEFINE&ARRAY#, string, 20
:FILL&ARRAY#[] = GET_VAR(VARA.TEST, KEY1)
:PRINT"Erster Wert der Variable VARA.TEST des Keys KEY1: &ARRAY#[1]"
```

Im zweiten Beispiel werden die Spalten der ersten Zeile einer Log-Datei (Automation Engine) in das Array angelegt.

```
:DEFINE &ARRAY#, string, 20
:SET &HND# = PREP_PROCESS_FILE(WIN01,
"C:\AUTOMIC\AUTOMATIONENGINE\TEMP\CPsrv_log_001_
00.txt","*","COL=LENGTH""LENGTH_TAB,,,='8=DATE,1,6=TIME,7,200=TEXT'")
:PROCESS &HND#
: FILL&ARRAY#[] = GET_PROCESS_LINE(&HND#)
:ENDPROCESS

:SET &RUNVAR# = 1
:SET &LEN# = LENGTH(&ARRAY#[])

:WHILE &RUNVAR# LE &LEN#
:P "Line &RUNVAR#: &ARRAY#[&RUNVAR#]"
:SET &RUNVAR# = &RUNVAR# + 1
:ENDWHILE
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:PUBLISH	Definiert Scriptvariablen und Arrays als Objektvariablen.
GET_PROCESS_LINE	Ermittelt den aktuellen Zeileninhalt einer Datensequenz.
GET_VAR	Liefert den Inhaltseintrag eines Variablen-Objektes.
FIND	Durchsucht ein Script-Array und liefert den entsprechenden Index zurück.
LENGTH	Ermittelt die Größe eines Script-Arrays.

Script-Sprachmittel - Datensequenzen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5.8 :GENERATE

Script-Anweisung: Steuert die Behandlung von Script-Zeilen bei der Prozessierung eines Scripts.

Syntax

:GEN[ERATE]*Generierungsmodus*

Syntaxteil	Beschreibung/Format

Generierungsmodus	Generierungsmodi für den Wiederanlauf: • ON_RESTART_ALWAYS Die Script-Zeilen werden, unabhängig vom Wiederanlauf-Punkt, immer durchgeführt. • ON_RESTART_CHECK Die Script-Zeilen werden nicht durchgeführt, wenn sie sich vor dem Wiederanlauf-Punkt befinden. • ON_RESTART_NEVER Die Script-Zeilen werden, unabhängig vom Wiederanlauf-Punkt, nie durchgeführt. Generierungsmodi für DATA-Zeilen: • UPPER_CASE Setzt den Text aller DATA-Zeilen auf Großbuchstaben um. • LOWER_CASE Der Text aller DATA-Zeilen wird auf Kleinbuchstaben umgesetzt. • CASE_UNCHANGED Stellt den Anfangszustand für die Groß- und Kleinschreibung wieder her. Es erfolgt keine weitere Umsetzung.

Anmerkungen

Der :GENERATE-Anweisung können verschiedene Ausdrücke als *Generierungsmodus* übergeben werden. Der *Generierungsmodus* steuert zum einen die Behandlung der Script-Zeilen beim Wiederanlauf eines aktivierbaren Objektes. Zum anderen kann damit die Groß- und Kleinschreibung in Data-Zeilen festgelegt werden.

Wiederanlauf von aktivierbaren Objekten

Beim Wiederanlauf von aktivierbaren Objekten kann mit dem *Generierungsmodus* festgelegt werden, wie Script-Zeilen bei der Prozessierung des Scripts zu behandeln sind. Diese Festlegung bleibt solange gültig bis:

- zur nächsten :GENERATE-Anweisung, die einen der drei Generierungsmodi zum Wiederanlauf enthält oder
- zur nächsten :RESTART-Anweisung.

Ohne der Verwendung von :GENERATE werden beim Wiederanlauf alle Script-Zeilen bis zur ersten :RESTART-Anweisung prozessiert und zusätzlich jene Zeilen, die sich nach dem angegebenen Wiederanlauf-Punkt befinden.

Groß- und Kleinschreibung

Generell wird der Text in den DATA-Zeilen eines Scripts so belassen und dem Zielsystem übergeben, wie er hinterlegt wurde. Eine Ausnahme bilden Script-Variablen, die in DATA-Zeilen vorkommen können. Diese werden bei der Aktivierung des Objektes, welches ein solches Script enthält, versorgt und damit geändert.

Bei manchen Zielsystemen kann es erforderlich sein, die DATA-Zeilen eines Scripts in einer bestimmten Schreibweise zu übermitteln. Beispielsweise verarbeitet das Betriebssystem BS2000 nur JCL (Job Control Language) in Großbuchstaben, so dass in der Automation Engine die Scripts entsprechend aufbereitet sein müssen. Mit der :GENERATE-Anweisung stellen wir Ihnen verschiedene Generierungsmodi zur Verfügung, damit Sie beim Aufbau Ihrer Scripts unabhängig von den Restriktionen des jeweiligen Zielsystems sind.

Beispiele

Das erste Beispiel zeigt die Anwendung beim Wiederanlauf des Objektes und die verschiedenen Ausgaben im Report, die aus den Wiederanlauf-Punkten resultieren.

```
:PRINT "Scriptbeginn"

:RESTART R1
:PRINT "Punkt R1"

:GENERATE ON_RESTART_ALWAYS
:PRINT "ON_RESTART_ALWAYS"

:GENERATE ON_RESTART_CHECK
:RESTART R2
:PRINT "Punkt R2"
:RESTART R3
:PRINT "Punkt R3"

:GENERATE ON_RESTART_NEVER
:PRINT "Scriptende"
```

Wiederanlauf mit R1:

```
2005-01-31 12:17:05 - U0020408 Scriptbeginn
2005-01-31 12:17:05 - U0020408 Punkt R1
2005-01-31 12:17:05 - U0020408 ON_RESTART_ALWAYS
2005-01-31 12:17:05 - U0020408 Punkt R2
2005-01-31 12:17:05 - U0020408 Punkt R3
```

Wiederanlauf mit R2:

```
2005-01-31 12:17:23 - U0020408 Scriptbeginn
2005-01-31 12:17:23 - U0020408 ON_RESTART_ALWAYS
2005-01-31 12:17:23 - U0020408 Punkt R2
2005-01-31 12:17:23 - U0020408 Punkt R3
```

Wiederanlauf mit R3:

```
2005-01-31 12:17:48 - U0020408 Scriptbeginn
2005-01-31 12:17:48 - U0020408 ON_RESTART_ALWAYS
2005-01-31 12:17:48 - U0020408 Punkt R3
```

Das zweite Beispiel setzt den Text der DATA-Zeilen eines Scripts in Großbuchstaben um.

```
:GEN UPPER_CASE
fs $automic.
```

Siehe auch:

:RESTART	Setzt Wiederanlauf-Punkte in einem aktivierbaren Objekt.

Script-Sprachmittel - Script-Aufbau und -Prozessierung
Wiederanlauf aktivierbarer Objekte
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5.9 :IF... :ELSE... :ENDIF

Script-Anweisungen: Werten einen Ausdruck aus und führen in Abhängigkeit vom Ergebnis Anweisungen aus.

Syntax

:IF *Bedingung*
[Anweisungen]
[:ELSE]
[Sonst-Anweisungen]
:ENDIF

Syntaxteil	Beschreibung/Format
Bedingung	Ein mittels Script-Literalen, Script-Variablen, Script-Funktionen oder numerischen Ausdrücken sowie Operatoren formulierter Ausdruck, der als Ergebnis "Wahr" oder "Falsch" liefert.
Anweisungen	Eine oder mehrere Anweisungen, die ausgeführt werden, wenn die *Bedingung* den Wert "Wahr" ergibt. Format: Script-Anweisung
Sonst-Anweisungen	Eine oder mehrere Anweisungen, die ausgeführt werden, wenn die *Bedingung* den Wert "Falsch" ergibt. Format: Script-Anweisung

Anmerkungen

Die IF-Anweisung ist eine Kontrollstruktur, mit der Sie die Scriptabarbeitung von dem Ergebnis einer *Bedingung* abhängig machen können. Bei dieser handelt es sich um einen logischen Ausdruck.

Ist die Bedingung erfüllt (Wahr), so werden die *Anweisungen* ausgeführt.

Ist die Bedingung nicht erfüllt (Falsch), so werden die *Sonst-Anweisungen* von :ELSE ausgeführt. Besitzt der IF-Block keinen :ELSE-Zweig, wird das Script mit der nächsten Zeile nach dem :ENDIF fortgesetzt.

Sie können IF-Blöcke auch schachteln, wobei es keine Limitierung der Tiefe gibt.

Ein IF-Block muss nicht unbedingt einen ELSE-Zweig beinhalten.

Verwenden eines OR-Schlüsselwortes

Um eine Bedingung mit mehreren Werten zu verknüpfen bzw. mehrere einzelne Bedingungen zu verknüpfen, dient das Schlüsselwort OR.

Eine *Bedingung* weist folgende Struktur auf:

*Wert**Vergleichsoperator**Vergleichswert[*OR *Vergleichswert...]*

Beispiel:

```
:IF &ABT# = "EDV" OR "automic"
```

Bis zu 12 OR dürfen in der Bedingung vorhanden sein.

Vergleichsoperatoren

Die Formulierung einer *Bedingung* besitzt eine der folgenden Vergleichsoperatoren:

Operator	Kürzel	Regel
=	EQ	Der Ausdruck ist "Wahr", wenn mindestens ein *Vergleichswert* dem *Wert* entspricht.
<>	NE	Der Ausdruck ist "Wahr", wenn keiner der *Vergleichswerte* gleich dem *Wert* ist.
<	LT	Der Ausdruck ist "Wahr", wenn ein *Vergleichswert* den angegebenen Begriff erfüllt.
>	GT	
<=	LE	
>=	GE	

Weisen beide Vergleichswerte der Bedingung einen numerischen Wert auf (Datentyp signed, unsigned oder float), so wird numerisch vergleichen. Ansonsten wird ein String-Vergleich gemacht. Bei einem numerischen Vergleich werden führende Nullen ignoriert, bei einem String-Vergleich nicht.

Die folgende Bedingung liefert "Wahr":

```
:IF"0" = "0"
```

Handelt es sich um keine numerischen Werte, so erfolgt ein alphanumerischer Vergleich.

Die folgende Bedingung liefert "Falsch":

```
:IF"0" = "abc"
```

Beim Vergleich von Zeichenfolgen wird nicht deren Länge geprüft. Die folgende Bedingung liefert "Falsch", da "A" im Alphabet vor "M" steht.

```
:IF "Meier" < "AUTOMIC"
```

Sie müssen die Script-Variablen, die in der *Bedingung* vorkommen, schon vorher definieren.

Bei Bedingungen, die mehrmals das Schlüsselwort OR enthalten, kann sich abhängig vom Operator ein unterschiedliches Verhalten ergeben.

Beispiel:
In diesem Fall muss einer der Werte zutreffen, damit die Bedingung erfüllt wird.

```
:IF&WD# = "MO" OR "DI" OR "MI"
```

Folgende Bedingung allerdings wird nur dann erfüllt, wenn die Variable keinem der drei Werte entspricht.

```
:IF&WD# <> "DO" OR "FR" OR "SA"
```

Abschließende Leerzeichen werden nicht berücksichtigt

Abschließende Leerzeichen werden in String-Vergleichen nicht berücksichtigt. Deshalb ist die Länge von leeren Strings nicht relevant.

Beispiel:

```
:set&a= 'a    '
:set&b= 'a'

:if&a = &b
:print 'true'
:endif
```

Ein Vergleich von **&a** mit **&b** würde in diesem Falle "true" ergeben, weil leere Strings vor dem Vergleich entfernt werden.

Beispiele

Diese Beispiele zeigen verschiedene Möglichkeiten zur Formulierung der *Bedingung*. Während die Beispiele 1 bis 3 eine Script-Variable mit einem oder mehreren *Vergleichswerten* in Beziehung setzen, wertet das 4. Beispiel den Rückgabewert einer Script-Funktion aus.

```
:IF&USER# = "TSOS"
!...
:ENDIF

:IF&DAT1# EQ &DAT2# OR " "
!...
:ENDIF

:IF&USER# <> "TSOS" OR "SERVICE" ORSYS_USER_NAME()
!...
:ENDIF

:IFGET_ATT(OO) = "J"
!...
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:WHILE... :ENDWHILE	Definition einer Schleife zur mehrfachen Ausführung von Script-Teilen.

Script-Sprachmittel - Script-Aufbau und -Prozessierung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5.10 :INCLUDE

Script-Anweisung: Bindet ein Include-Objekt in ein Script ein.

Syntax

:INC[LUDE] ***Include-Objekt*** *[StringAlt = StringNeu][,***NOFOUND=IGNORE***][,***EXT_REPORT=OFF***]**

Syntaxteil	Beschreibung/Format
Include-Objekt	Name des einzubindenden Include-Objektes. Geben Sie den vollständigen Namen des Include-Objektes an. Script-Variablen sind nicht erlaubt!
StringAlt	Zeichenfolge aus dem Script des Include-Objektes, die durch *StringNeu* ersetzt werden soll. Format: Script-Literal
StringNeu	Zeichenfolge, die *StringAlt* aus dem Script des Include-Objektes ersetzt. Format: Script-Literal maximal 50 Zeichen
NOFOUND=IGNORE	Wird das angegebene Include-Objekt nicht gefunden, entsteht kein Fehler.
EXT_REPORT=OFF	Der Inhalt des Include-Objektes wird nicht in den erweiterten Reports protokolliert. Es wird stattdessen eine Hinweiszeile ausgegeben.

Anmerkungen

Viele Objekte in der Automation Engine verwenden in ihrem Script gleiche Verarbeitungsschritte. Um diese nicht in jedem einzelnen Script schreiben und pflegen zu müssen, gibt es die Include-Objekte. Diese enthalten in ihrem Script häufig wiederkehrende und oft verwendete Script-Teile.

Um ein Include-Objekt aus einem anderen Objekt heraus ansprechen zu können, wird die :INCLUDE-Anweisung verwendet. Bei der Aktivierung des Objektes wird das Script aus dem Include-Objekt hinzugeneriert.

StringAlt und *StringNeu* sind optional. Sie können dazu verwendet werden, um eine Zeichenfolge aus dem Include-Objekt durch eine neue zu ersetzen. Diese Zuweisung ist nur für die laufende Generierung gültig und ändert das Include-Objekt selbst nicht.

Beispiele

Das Beispiel zeigt die Einbindung eines Include-Objektes. Dabei wird anstatt der im Include-Objekt angegebenen Zeichenfolge "$MAWI." eine neue namens "$MAWITEST." zugewiesen.

```
:INC MAWI.DATEIZUWEISUNGEN "$MAWI." = "$MAWITEST."
```

Im Beispiel wird ein Benutzer-Include aufgerufen. Ist dieses nicht vorhanden, entsteht kein Fehler.

```
:INCLUDE HEADER.WINDOWS.USER.HEAD ,NOFOUND=IGNORE
```

Im dritten Beispiel wird unterdrückt, dass der Inhalt des Include-Objektes in den erweiterten Reports protokolliert wird.

```
:INC MAWI.DATEIZUWEISUNGEN ,EXT_REPORT=OFF
```

Siehe auch:

:INC_SCRIPT	Bindet ein Script in ein anderes Script des gleichen Objektes ein.

Script-Sprachmittel - Script-Aufbau und -Prozessierung

Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5.11 :INC_SCRIPT

Script-Anweisung: Bindet ein Script in ein anderes Script des gleichen Objektes ein.

Syntax

:INC_SCRIPT ***(Script-ID)***

Syntaxteil	Beschreibung/Format
Script-ID	Nummer, mit der das Script eines Objektes identifiziert wird. Format: Zahl ohne Hochkommata Erlaubte Werte: "0", "1" und "2" "0" - Bindet das Script ein (nur bei Jobs und Ereignissen) "1" - Bindet das Pre-Script ein (nur bei Jobs), bindet das !Script ein (nur bei Ereignissen) "2" - Bindet das Post-Script ein (nur bei Jobs)

Anmerkungen

Diese Script-Anweisung wurde aus internen Gründen implementiert, damit ein Script im Header aufgerufen werden kann. In diesem Script können bestimmte Attribute gesetzt werden, welche dann im Header benötigt werden (beispielsweise ein anderer Host). Mit diesem Script können auch spezielle Anforderungen der z/OS-JCL realisiert werden.

Die Script-Anweisung könnte aber auch "normal" genutzt werden, um ein Script in einem anderen Script - auch mehrmals - aufzurufen. Beide Scripts müssen zum gleichen Objekt gehören. Daher ist die Script-Anweisung derzeit nur in Jobs oder Ereignissen möglich. Nur diese haben mehr als eine Registerkarte für Scripts.

Um Ihr Script modular aufzubauen, empfehlen wir Ihnen die Verwendung der Script-Anweisung :INCLUDE.

Beispiel

Die Registerkarte **Pre-Script** eines Jobs enthält einen DIR-Befehl. In der Registerkarte **Script** wird das Pre-Script eingebunden.

```
:INC_SCRIPT (1)
```

Im Job-Report können Sie anschließend erkennen, dass der DIR-Befehl zweimal ausgeführt wurde. Das erste Mal im Header (definiert in der Registerkarte "Pre-Script"). Das zweite Mal zwischen Header und Trailer (durch die Script-Anweisung in der Registerkarte **Script**).

Siehe auch:

Script-Sprachmittel	Beschreibung
:INCLUDE	Bindet ein Include-Objekt in ein Script ein.

Script-Sprachmittel - Script-Aufbau und -Prozessierung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5.12 :JCL_CONCAT_CHAR

Script-Anweisung: Bildet JCL-Zeilen bis zu einer maximalen Länge von 2 KB.

Syntax

:JCL_CONCAT_CHAR *[Zeichen]*

Syntaxteil	Beschreibung/Format
Zeichen	Einzelnes Zeichen das als Verbindungszeichen für die nachfolgende JCL-Zeile agiert. Format: Script-Literal oder Script-Variable

Anmerkungen

Mit der Script-Anweisung können mehrere Zeilen eines Scripts, welche JCL enthalten, zu einer einzigen JCL-Zeile verbunden werden. Diese Verarbeitung erfolgt, wenn der Job generiert wird. Für JCL-Zeilen gibt es keine Beschränkung. Die maximale Länge einer üblichen Scriptzeile ist hingegen auf 1024 Zeichen beschränkt.

Scriptzeilen, welche eine JCL-Zeile bilden sollen, enden mit dem angegebenen *Zeichen*. Die Scriptzeilen werden an dieser Stelle miteinander verbunden. Die letzte Scriptzeile, die zur JCL-Zeile gehört, endet ohne das *Zeichen*.

Das *Zeichen* selbst ist in der gebildeten JCL-Zeile nicht enthalten. Leerzeichen, die in der Scriptzeile unmittelbar vor dem *Zeichen* stehen, werden in die JCL-Zeile übernommen. Leerzeichen, mit denen Scriptzeilen beginnen, werden eliminiert. Dadurch ist es möglich, einzelne JCL-Anweisungen einzurücken. Das erhöht die Lesbarkeit des Scripts.

Es können mehrere JCL-Zeilen nacheinander gebildet werden. Der erneute Aufruf der Script-Anweisung ohne Parameter beendet diese Verarbeitung von Scriptzeilen.

Wird bei der Verwendung des Script-Sprachmittels kein Zeichen angegeben, ist es nicht möglich mehrere JCL-Zeilen zu verbinden. Die Script-Anweisung besitzt in diesem Fall keine Funktion.

Beachten Sie, dass nicht mehrere Kommandos zusammengefasst werden können! Das Script-Sprachmittel kann nur Zeilen verbinden, die zu einem Kommando gehören!

JCL-Zeilen, welche über die Forms-Registerkarte von SAP, JMX, PeopleSoft und Datenbank-Jobs eingefügt werden und eine gewisse Länge aufweisen, werden automatisch durch dieses Sprachmittel in mehrere Zeilen aufgeteilt. In der Registerkarte **Script** dieser Jobs sollte daher das Sprachmittel nicht manuell eingefügt werden, wenn Sie die Forms-Registerkarte verwenden, da dies mitunter zu Problemen führen kann!

Beispiel

Im Beispiel wird Stern (*) als das Zeichen definiert, über welches die einzelnen Scriptzeilen verbunden werden sollen. Danach erfolgt der Aufruf eines Dienstprogramms des MS SQL Servers mit einigen Parametern. Der Parameter EXIT, welcher ein SQL Statement enthält, steht in einer neuen Scriptzeile. Das SQL Statement selbst wurde auf mehrere Scriptzeilen aufgeteilt.

```
:JCL_CONCAT_CHAR  "*"
ISQL.EXE /U SA /P /S BUNTWS02 /t 10 /d UC4 /Q *
"EXIT(UPDATE OH SET OH_EXPKZ = 1 *
WHERE OH_MANDANT = 01 *
AND (OH_OTYP_TYP <> 'FOLD') *
AND OH_LOEKZ = 0)"
:JCL_CONCAT_CHAR
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:JCL_SUBSTITUTE	Ersetzt eine Zeichenkette in der JCL durch eine andere Zeichenkette.

Script-Sprachmittel - Script-Aufbau und -Prozessierung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5.13 :JCL_SUBSTITUTE

Script-Anweisung: Ersetzt eine Zeichenkette in der JCL durch eine andere Zeichenkette.

Syntax

:JCL_SUBSTITUTE *[Alte Zeichenkette, Neue Zeichenkette]*

Syntaxteil	Beschreibung/Format
Alte Zeichenkette	Zeichenkette, die ersetzt werden soll. Format: Script-Literal oder Script-Variable
Neue Zeichenkette	Zeichenkette, die *Alte Zeichenkette* ersetzt. Format: Script-Literal oder Script-Variable

Anmerkungen

Die Script-Anweisung ersetzt bei der Generierung eines Jobs eine beliebige Zeichenkette in der JCL (Job Control Language). Die Anzahl der Zeichen in *Alte Zeichenkette* und *Neue Zeichenkette* kann unterschiedlich sein. Es kann auch nur ein einzelnes Zeichen ersetzt werden.

Die Script-Anweisung dient beispielsweise dazu, spezielle Anforderungen der z/OS-JCL zu erfüllen. Dort wird das Zeichen "&" benötigt. Da dieses Zeichen in der AE zur Definition von Script-Variablen verwendet wird, können Konflikte auftreten.

Die angegebenen Zeichenketten werden ab der Zeile mit der Script-Anweisung ersetzt. Sie werden bis zu der Zeile des Scripts ersetzt, in der die Script-Anweisung mit oder ohne Parameter erneut aufgerufen wird.

Beispiel

Im Beispiel wird das Zeichen "$" durch das Zeichen "&" ersetzt.

```
:JCL_SUBSTITUTE  "$", "&"
REPORTS=($REP)
SORT=TIME
:JCL_SUBSTITUTE
RANGE=$RAN
```

Der generierte Job enthält dadurch folgende Zeilen:

```
REPORTS=(&REP)
SORT=TIME
RANGE=$RAN
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:JCL_CONCAT_CHAR	Bildet JCL-Zeilen bis zu einer maximalen Länge von 2 KB.

Script-Sprachmittel - Script-Aufbau und -Prozessierung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5.14 :PSET

Script-Anweisung: Weist einer Objektvariablen einen Wert zu.

Syntax

:PSET *Objektvariable = Wert*

Syntaxteil	Beschreibung/Format
Objectvariable	Name der Objektvariable mit vorangestelltem "&", die mit einem Wert versorgt werden soll. Ein Script-Variablenname für die Script-Sprachmittel :PSET und :RSET ist beschränkt auf 31 alphanumerische Zeichen (meistens sind die Script-Variablennamen auf 32 Zeichen beschränkt). Es dürfen die Sonderzeichen "$", "_", "@", "§" und "#" verwendet werden. Deutsche Umlaute nicht sind erlaubt und das erste Zeichen darf keine Nummer sein. Variablen innerhalb des Scripts müssen immer mit einem führenden "&"beginnen, dem der Variablenname folgt. Format: Script-Variable

Wert	Wert, welcher der Objektvariablen zugewiesen wird. Format: Script-Literal, Script-Variable oder Script-Funktion

Anmerkungen

Objektvariablen werden in der Registerkarte "Variablen & Prompts" des Objektes hinterlegt.

Der Workflow in dem sich der Job befindet, innerhalb dessen :PSET verwendet wird, ist Processor des Jobs. Er wird auch als Parent bezeichnet.

Es folgen die einzelnen Schritte, die :PSET durchführt:

1. Ersetzung bzw. Hinzufügung der Objektvariable in der Aufgabe selbst.
2. Ersetzung bzw. Hinzufügung der Objektvariable im Parent.

Der geänderte Wert bzw. die zusätzliche Objektvariable gilt nur für die Durchführung der Aufgaben. Im Objekt selbst wird der Wert nicht dauerhaft gespeichert.

Beachten Sie, dass die Einstellung "Zur Laufzeit generieren" in den Objekten großen Einfluss hat! Wenn das Script bereits fertig generiert worden ist, wirken sich spätere Änderungen an den Objektvariablen nicht mehr auf dessen Inhalt aus.

Beachten Sie, dass Aufgaben Objektvariablen nicht zwangsläufig übernehmen. In jedem Objekt ist einstellbar ob und welche Objektvariablen von der übergeordneten Aufgabe geerbt werden sollen.

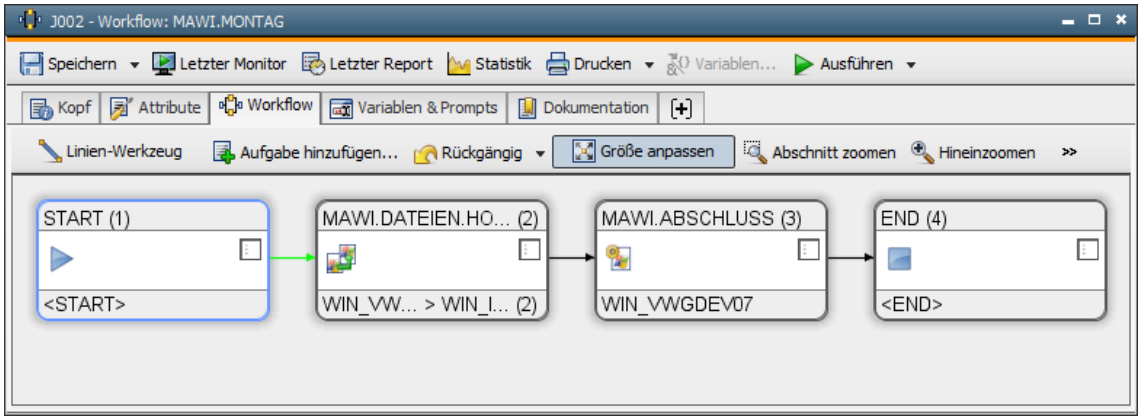
Beachten Sie, dass :PSET die Objektvariablen in verschachtelten Workflows nicht bis zum obersten Workflow weiterreicht. Die Objektvariablen werden nur im Workflow ersetzt bzw. hinzugefügt in dem sich die Aufgabe befindet, welche die Script-Anweisung aufruft.

Beachten Sie, dass geerbte Objektvariablen, die nicht in der Aufgabe definiert wurden, beim Wiederanlauf nur dann zur Verfügung stehen, wenn sich die Aufgabe noch im Aktivitätenfenster befindet! Wenn für den Job im folgenden Beispiel ein Wiederanlauf durchgeführt werden würde, so kann er nur dann auf die Objektvariable &HOST# zugreifen, wenn sich der FileTransfer und der Workflow noch im Aktivitätenfenster befinden.

Beachten Sie außerdem, dass ein Script, in dem `ACTIVATE_UC_OBJECT` als Aktivator des Jobs genutzt wird, nicht Parent (übergeordnete Aufgabe) des Jobs ist. Ein Abbruch des Scripts wirkt sich nicht auf den Job aus.
Dies steht im Gegensatz zum Abbruch des Jobs im Workflow. Dieser Abbruch wirkt sich durch die Parent-Child-Beziehung zwischen Workflow und Job auf den Job direkt aus.

Beispiel

Ein Workflow enthält zwei Objekte namens "MAWI.DATEIEN.HOLEN" und "MAWI.ABSCHLUSS". Beide haben die Einstellung "Zur Laufzeit generieren" gesetzt, damit ihr Script erst erzeugt wird, wenn sie im Workflow an der Reihe sind.



Der FileTransfer prüft die Umgebung und ändert bei Bedarf die Objektvariable &HOST#.

```
:PSET &HOST# = "unix01"
```

Der nachfolgende Job verwendet diese Objektvariable in seinem Script. Eine Änderung mit :PSET wirkt sich daher auch auf ihn aus, da er alle Objektvariablen seines Parents erbt.

Siehe auch:

Script-Sprachmittel	Beschreibung
:RSET	Weist einer Script-Variablen einen Wert zu und speichert diesen im Aktivierungsreport.
:SET	Weist einer Script-Variablen einen Wert zu.
:SET_SCRIPT_VAR	Setzt die Werte von Script-Variablen per indirektem Zugriff.

Script-Sprachmittel - Script-Aufbau und -Prozessierung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5.15 :PUBLISH

Script-Anweisung: Script-Variablen und Arrays als Objektvariablen definieren.

Syntax

:PUB[LISH] ***Variablenname*** *[, [Objektvariable] [,Gültigkeitsbereich]]*

Syntaxteil	Beschreibung/Format

Variablenname	Name der Script-Variable bzw. des Script-Arrays (inklusive &). Der Name einer Script-Variablen besteht aus maximal 32 alphanumerischen Zeichen einschließlich der Sonderzeichen "$", "_", "@", "§" und "#". Umlaute sind nicht erlaubt. Das erste Zeichen des Namens darf keine Zahl sein. Im Script sind Variablen immer mit dem Kennzeichen "&" vor dem Namen anzugeben! Format: Script-Variable Bei Arrays sind am Ende des Namens zusätzlich die leeren Indexklammern [ ] anzugeben.
Objektvariable	Neuer Name für die Objektvariable (ohne führendes &) Format: AE-Name Bei Script-Arrays sind die Indexklammern [] nicht anzugeben!
Gültigkeitsbereich	Legt fest, an welchen Aufgaben-Bereich die Variable weitergegeben werden soll. Erlaubte Werte: "TASK", "WORKFLOW" oder "TOP" "TASK" - Objektvariable steht nur der eigenen Aufgabe zur Verfügung "WORKFLOW" - Objektvariable an den Parent-Workflow weitergeben "TOP" - Objektvariable an den obersten Workflow weitergeben

Anmerkungen

Die Anweisung :PUBLISH wandelt eine Script-Variable bzw. ein Script-Array in eine Objektvariable um und gibt diese, wenn gewünscht, an übergeordnete Aufgaben weiter.

Wird der Parameter *Objektvariable* spezifiziert, so wird eine neue Objektvariable mit diesem Namen und dem Wert der Script-Variablen angelegt. Existiert bereits eine Objektvariable mit diesem Namen, so wird diese überschrieben. Die Script-Variable bleibt in diesem Fall unverändert.

⚠ Beachten Sie, dass die angegebene Script-Variable bzw. das Array existieren muss! Andernfalls kommt es zu einem Laufzeitfehler.

Wird ein Script-Array angegeben, so werden die leeren Elemente am Ende des Arrays für die Objektvariable nicht übernommen bzw. entfernt. Dies trifft sowohl bei Umwandung der Script-Variable als auch bei Erstellung einer neuen Objektvariable zu.

⚠ Wenn ein Script-Array angegeben wird, welches sehr viele Elemente (Bereich 10000 oder mehr) aufweist, so kann dies zu Performance-Einbussen führen.

Die Script-Anweisung ähnelt dem Sprachmittel :PSET, unterscheidet sich jedoch dadurch, dass die angegebene Script-Variable existieren muss und auch Arrays angegeben werden können.

Zusätzlich können Sie mit dem Parameter *Gültigkeitsbereich* festlegen, welchen Aufgaben die Objektvariable zur Verfügung stehen soll. Die Variable kann dabei an den Parent-Workflow, dem Top-Workflow weitergegeben werden oder nur der eigenen Aufgabe zugänglich gemacht werden.

Beispiele

In folgendem Beispiel wird der Objektname in eine Script-Variable gespeichert und diese mittels :PUBLISH an den übergeordneten Workflow weitergegeben.

```
:SET &CHILD# = SYS_ACT_ME_NAME()
:PUBLISH &CHILD#,,"WORKFLOW"
```

Im zweiten Beispiel wird ein Script-Array erstellt und mit den Werten eines Variablen-Objektes befüllt. Anschließend wird das Array als neue Objektvariable an den obersten Workflow weitergegeben.

```
:DEFINE &ARRAY#, string, 5
:FILL &ARRAY#[] = GET_VAR(VARA.TEST, KEY1)
:PUBLISH &ARRAY#[], VARA_ARRAY#, "TOP"
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:FILL	Speichert mehrere Werte in ein Script-Array.
GET_PUBLISHED_VALUE	Wert einer Objekt- bzw. PromptSet-Variablen einer bestimmten Aufgabe ermitteln.

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5.16 :RESTART

Script-Anweisung: Setzt Wiederanlauf-Punkte in einem aktivierbaren Objekt.

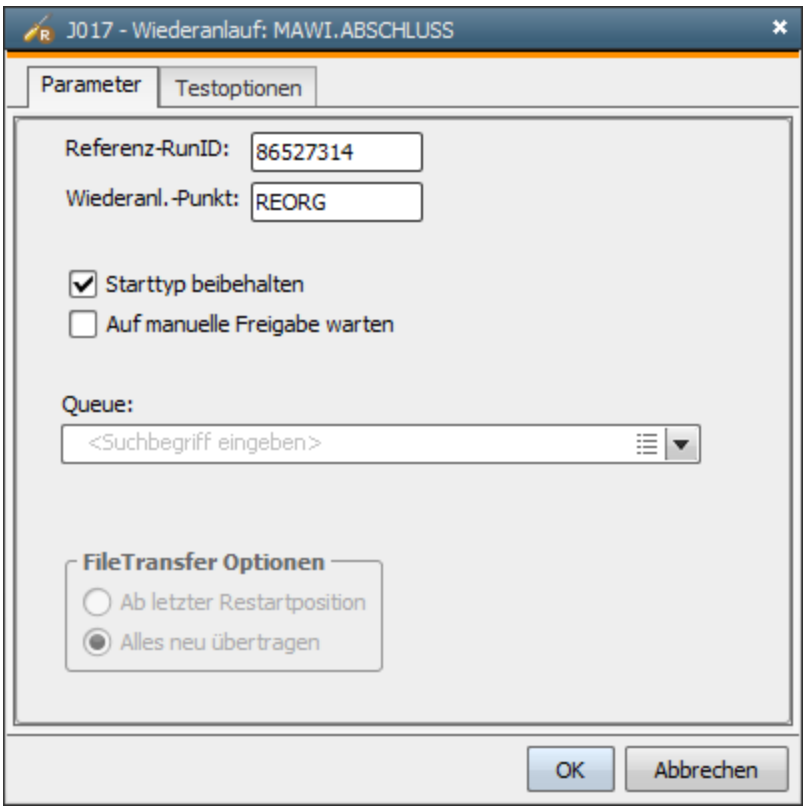
Syntax

:RESTART*Wiederanlauf-Punkt* *[, Wiederanlauf-Text]*

Syntaxteil	Beschreibung/Format
Wiederanlauf-Punkt	Bezeichnung des Wiederanlauf-Punkts. Format: Wiederanlauf-Punkt-Bezeichnung ohne Hochkommata Maximal 8 Zeichen
Wiederanlauf-Text	Beschreibung des Wiederanlauf-Punktes. Format: Script-Literal oder Script-Variable Maximal 24 Zeichen

Anmerkungen

Mit der :RESTART-Anweisung können in einem aktivierbaren Objekt Wiederanlauf-Punkte gesetzt werden. Wiederanlauf-Punkte erlauben Ihnen, ein aktivierbares Objekt ab einem bestimmten Schritt auszuführen. Dazu muss beim Ausführen mit Optionen der gewünschte *Wiederanlauf-Punkt* angegeben werden.



Mit dem *Wiederanlauf-Text* legen Sie eine Beschreibung des Wiederanlauf-Punktes fest, die in den Durchführungsprotokollen als zusätzliche Information ausgegeben wird. Die Angabe des Wiederanlauf-Textes ist optional.

⚠ In welcher Weise das Script beim Wiederanlauf behandelt wird, hängt von der :GENERATE-Anweisung ab! Verwenden Sie dieses Script-Sprachmittel nicht, so werden alle Scriptzeilen bis zur allerersten :RESTART-Anweisung durchgeführt und zusätzlich jene Zeilen die sich nach dem angegebenen Wiederanlauf-Punkt befinden.

Den zuletzt durchlaufenen Wiederanlauf-Punkt samt Wiederanlauf-Text können Sie mit den Scriptfunktionen SYS_LAST_RESTART_POINT bzw. SYS_LAST_RESTART_TEXT auslesen.

Beispiel

Das Beispiel definiert einen *Wiederanlauf-Punkt* namens "REORG" mit *Wiederanlauf-Text*.

```
:RESTART REORG, "Start Reorganization"
```

Siehe auch:

:GENERATE	Steuert die Behandlung von Script-Zeilen bei der Prozessierung eines Scripts.
RESTART_UC_ OBJECT	Wiederholt die Durchführung einer Aufgabe.
SYS_ACT_RESTART	Ermittelt, ob Objekt im Wiederanlauf-Modus aktiviert wurde.
SYS_ACT_RESTART_ ME_NR	Liefert die laufende Nummer (RunID) des im Wiederanlauf-Modus aktivierten Objektes.
SYS_LAST_RESTART_ POINT	Liefert die Bezeichnung des vorangegangenen Wiederanlauf-Punktes im Script.
SYS_LAST_RESTART_ TEXT	Liefert den Text des vorangegangenen Wiederanlauf-Punktes im Script.
SYS_RESTART_POINT	Liefert den Wiederanlauf-Punkt, mit dem das Objekt ausgeführt wird.

Script-Sprachmittel - Script-Aufbau und -Prozessierung
Wiederanlauf aktivierbarer Objekte
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5.17 :RSET

Script-Anweisung: Weist einer Script-Variablen einen Wert zu und speichert diesen im Aktivierungsreport.

Syntax

:RSET*Script-Variable* *[= Wert]*

Syntaxteil	Beschreibung/Format
Script-Variable	Name der Script-Variable, die mit einem Wert versorgt werden soll. Ein Script-Variablenname für die Script-Sprachmittel :PSET und :RSET ist beschränkt auf 31 alphanumerische Zeichen (meistens sind die Script-Variablennamen auf 32 Zeichen beschränkt). Es dürfen die Sonderzeichen "$", "_", "@", "§" und "#" verwendet werden. Deutsche Umlaute nicht sind erlaubt und das erste Zeichen darf keine Nummer sein. Variablen innerhalb des Scripts müssen immer mit einem führenden "&"beginnen, dem der Variablenname folgt. Format: Script-Variable
Wert	Wert, welcher der Script-Variable zugewiesen wird. Format: Script-Literal, Zahl, Script-Variable oder Script-Funktion

Anmerkungen

Die Script-Anweisung speichert den Wert, den sie einer Script-Variablen zuweist, gleichzeitig im Aktivierungsreport. Dabei wird bei der Wertzuweisung folgende Zeile in das Aktivierunsprotokoll geschrieben:

```
JJJJ-MM-TT HH:MM:SS - U0020206 Variable '&VAR#' wurde mit Wert 'WERT'
gespeichert.
```

Neben dem Scriptsprachmittel :RSET ist es auch mit :READ möglich, Werte von Script-Variablen im Report festzuhalten.

Der gespeicherte Wert kann auf verschiedene Weise verwendet werden.

Zum einen wird ein Wiederanlauf der Aufgabe mit dem gespeicherten Wert durchgeführt. Die mit der Script-Anweisung verbundene Zuweisung des Wertes, wird im Wiederanlauffall nicht ausgeführt. Ist diese Script-Variable nicht auffindbar, bricht die Aktivierung ab. Wurde die Script-Variable mehrfach gespeichert, so wird der zuletzt zugewiesene Wert ohne Warnung genommen.

Zum anderen kann die Script-Anweisung auch dazu verwendet werden, um den Inhalt einer Script-Variablen bei Jobs zwischen Script und Post-Script zu übergeben. In der Registerkarte **Script** wird der Script-Variablen ein Wert zugewiesen. In der Registerkarte **Post-Script** wird die Script-Anweisung erneut aufgerufen, ohne aber einen Wert zuzuweisen. Die Script-Variable enthält dann den Wert aus dem Script, der als letztes im Aktivierungsreport gespeichert wurde.

⚠ Beachten Sie, dass bei der Speicherung des Wertes einer Script-Variable im Aktivierungsreport mit dem Scriptsprachmittel :RSET, die Datentypen "signed" und "float" automatisch zu "string" umgewandelt werden. Weist die Variable den Datentyp "unsigned" auf, so bleibt dieser erhalten.

⚠ Ein :RSET im Script von Workflows führt zur Erzeugung einer Objektvariable, die auch an untergeordnete Workflow-Aufgaben vererbt wird (sofern die Vererbung in den Child-Aufgaben aktiviert ist). Überschneidet sich der Name dieser Objektvariable mit jenen der Workflow-Aufgaben, so kann es zum Fehler bei deren Generierung kommen (siehe: Script-Variable - Syntax). Achten Sie daher darauf, Variablennamen immer mit einem Sonderzeichen abzuschließen, um Überschneidungen zu vermeiden.

⚠ Wenn Sie mit den Scriptsprachmitteln :SET und :RSET die gleiche Scriptvariable mit unterschiedlichen Werten versorgen, dann werden diese Werte sowohl mit :SET, als auch mit :RSET gesetzt, wenn der zugehörige Job ausgeführt wird. Bei einem Wiederanlauf dieses Jobs werden die Werte von beiden Variablen jedoch durch das Scriptsprachmittel :SET gesetzt, weil :RSET automatisch Objektwerte erstellt und :SET automatisch den Wert speichert für den Fall eines Wiederanlaufes. Dies ist ein geändertes Verhalten im Vergleich zu v8, wo :RSET noch keine Objektwerte erstellt hat.

Beispiel

Im Beispiel wird in der Registerkarte **Post-Script** auf eine Script-Variable zugegriffen, deren Wert in der Registerkarte **Script** gesetzt wurde.

Registerkarte Script:

```
:RSET  &TEXT# = "test"

:SET  &ZAHL# = 1
:RSET  &ZAHL# = ADD(&ZAHL#,1)
```

Registerkarte "Postscript":

```
:RSET  &TEXT#
:RSET  &ZAHL#

:PRINT  &TEXT#
:PRINT  &ZAHL#
```

Die Werte "test" und "2" werden an die Registerkarte "Postscript" übergeben und ins Aktivierungsprotokoll geschrieben.

Siehe auch:

Script-Sprachmittel	Beschreibung
:PSET	Weist einer Objektvariablen einen Wert zu.
:SET	Weist einer Script-Variablen einen Wert zu.
:SET_SCRIPT_VAR	Setzt die Werte von Script-Variablen per indirektem Zugriff.

Script-Sprachmittel - Script-Aufbau und -Prozessierung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5.18 :SET

Script-Anweisung: Weist einer Script-Variablen einen Wert zu.

Syntax

:S[ET] ***Script-Variable = Wert***

Syntaxteil	Beschreibung/Format
Script-Variable	Name der Script-Variable, die mit einem Wert versorgt werden soll. Der Name einer Script-Variablen besteht aus maximal 32 alphanumerischen Zeichen einschließlich der Sonderzeichen "$", "_", "@", "§" und "#". Umlaute sind nicht erlaubt. Das erste Zeichen des Namens darf keine Zahl sein. Im Script sind Variablen immer mit dem Kennzeichen "&" vor dem Namen anzugeben! Format: Script-Variable
Wert	Wert, welcher der Script-Variable zugewiesen wird. Format: Script-Literal, Script-Variable oder Script-Funktion

Anmerkungen

Beachten Sie die Besonderheiten im Umgang mit Script-Variablen.

Existiert die angegebene Variable noch nicht, so wird sie neu erstellt. In diesem Fall besitzt die Variable keinen bestimmten Datentyp.

Mit dem Scriptsprachmittel können auch arithmetische Terme aufgelöst werden. Mehr dazu siehe: Rechnen.

Werden vordefinierte Variablen in Klammern ( ) als *Wert* angegeben, die numerische Werte liefern (z.B.: &$CLIENT#), so werden diese automatisch in das 16-stellige Standardformat umgewandelt.

⚠ Wenn Sie mit den Scriptsprachmitteln :SET und :RSET die gleiche Scriptvariable mit unterschiedlichen Werten versorgen, dann werden diese Werte sowohl mit :SET, als auch mit :RSET gesetzt, wenn der zugehörige Job ausgeführt wird. Bei einem Wiederanlauf dieses Jobs werden die Werte von beiden Variablen jedoch durch das Scriptsprachmittel :SET gesetzt, weil :RSET automatisch Objektwerte erstellt und :SET automatisch den Wert speichert für den Fall eines Wiederanlaufes. Dies ist ein geändertes Verhalten im Vergleich zu v8, wo :RSET noch keine Objektwerte erstellt hat.

Beispiele

Das Beispiel zeigt die Zuweisung eines Dateinamens zur Script-Variablen "&DATEI#".

```
:SET &DATEI# = "L.LST.DATEI"
```

Im Beispiel wird der Script-Variablen das aktuelle Tagesdatum zugewiesen, welches von einer Script-Funktion zurückgeliefert wird.

```
:SET &HEUTE# = SYS_DATE(YYMMDD)
```

Natürlich kann eine Script-Variable auch Zahlen aufnehmen.

```
:SET &ANZAHL# = 1
```

Die Zuweisung eines Wertes einer anderen Script-Variable ist ebenfalls möglich.

```
:SET &NR# = &ANZAHL#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:DEFINE	Deklariert eine Script-Variable mit einem bestimmten Datentyp.
:PSET	Weist einer Objektvariablen einen Wert zu.
:RSET	Weist einer Script-Variablen einen Wert zu und speichert diesen im Aktivierungsreport.
:SET_SCRIPT_VAR	Setzt die Werte von Script-Variablen per indirektem Zugriff.

Script-Sprachmittel - Script-Aufbau und -Prozessierung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5.19 :SET_SCRIPT_VAR

Script-Anweisung: Setzt die Werte von Script-Variablen per indirektem Zugriff.

Syntax

:SET_SCRIPT_VAR ***Script-Variable = Wert***

Syntaxteil	Beschreibung/Format
Script-Variable	Platzhalter für die Namen von Script-Variablen, die mit Werten versorgt werden sollen. Der Name einer Script-Variablen besteht aus maximal 32 alphanumerischen Zeichen einschließlich der Sonderzeichen "$", "_", "@", "§" und "#". Umlaute sind nicht erlaubt. Das erste Zeichen des Namens darf keine Zahl sein. Im Script sind Variablen immer mit dem Kennzeichen "&" vor dem Namen anzugeben! Format: Script-Literal oder Script-Variable
Wert	Wert, welcher der Script-Variable zugewiesen wird. Format: Script-Literal, Script-Variable oder Script-Funktion

Anmerkungen

Die Script-Anweisung setzt die Werte von Script-Variablen, ohne dass dabei die Namen der Script-Variablen explizit angegeben werden müssen. Auf die Script-Variablen wird indirekt über einen Platzhalter, man könnte ihn selbst eine Variable nennen, zugegriffen. Dadurch ist es möglich, komfortabel die Werte vieler Script-Variablen, beispielsweise in einer Prozessschleife, zu setzen. Die Script-Anweisung ersetzt so viele bisher notwendige Bedingungsanweisungen durch eine einzige Script-Zeile.

Der Script-Anweisung wird mit *Script-Variable* eine Zeichenkette übergeben, aus denen der jeweilige Name einer Script-Variable gebildet wird. Die Zeichenkette darf dabei nicht mit einem "&" beginnen, wodurch eine Script-Variable gekennzeichnet ist. Es sind minimal nur soviele Anfangsbuchstaben des Script-Variablennamens (ohne &) anzugeben, die notwendig sind, um die Script-Variable eindeutig zu identifizieren. Die Script-Anweisung erstellt einen gültigen Namen für die Script-Variable und weist ihr *Wert* zu.

Ein Fehler tritt auf, wenn auf eine Script-Variable zugegriffen werden soll, die nicht existiert.

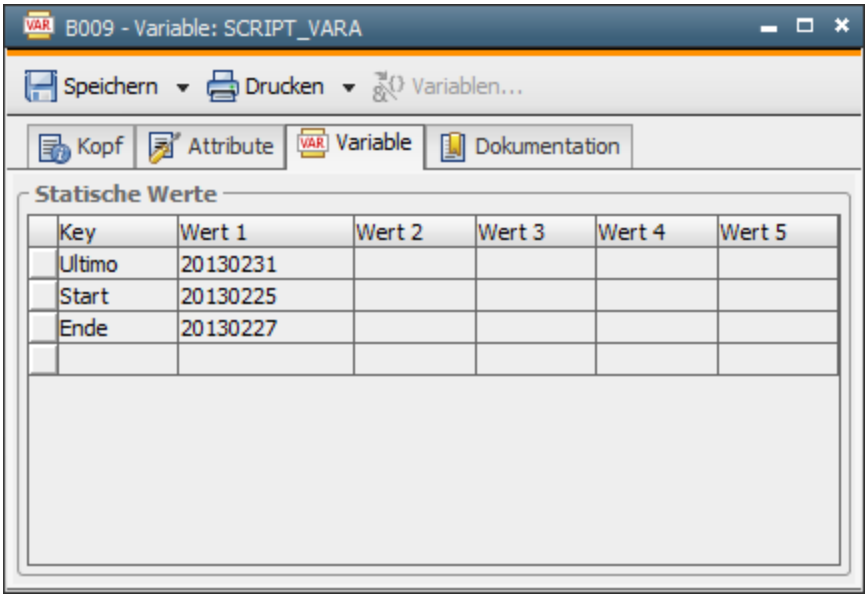
Trifft die angegebene Zeichenkette auf den Namen mehrerer Script-Variablen zu, so erfolgt keine Wertzuweisung. Wählen Sie die Zeichenkette daher so, dass die Script-Variable dadurch eindeutig identifiziert werden kann.

Die Script-Anweisung kann nicht dazu verwendet werden, Script-Variablen anzulegen. Diese müssen zuvor im Script mit der :SET-Anweisung erstellt und mit einem Initialwert versorgt worden sein.

⚠ Beachten Sie bei der Verwendung des Sprachmittels, dass beim Namen von Script-Variablen nicht zwischen Groß- und Kleinschreibung unterschieden wird. Dies bedeutet: Geben Sie Platzhalter (*Script-Variable*) an, die sich nur durch die Schreibweise unterscheiden, so wird immer auf die selbe Script-Variable zugegriffen.

Beispiel

Das Beispiel basiert auf einem Variablen-Objekt und einem Job. Im Variablen-Objekt sind die Namen von Script-Variablen (Key) und ihre Werte (Wert 1) gespeichert.



Im Script des Jobs wird in einer Prozessschleife zunächst der Name der Script-Variablen aus der Key-Spalte des Variablen-Objektes gelesen. Danach wird aus der 2. Spalte der Wert der Script-Variable gelesen. Anschließend werden die ermittelten Werte im Script den Variablen "&ENDE#", "&START#" und "&ULTIMO#" zugewiesen.

```
:SET   &ENDE#    = ""
:SET   &START#   = ""
:SET   &ULTIMO# = ""

:SET   &HANDLE# = PREP_PROCESS_VAR("SCRIPT_VARA")

:PROCESS   &HANDLE#
:    SET   &VARIABLE# = GET_PROCESS_LINE(&HANDLE#,1)
:    SET   &WERT#    = GET_PROCESS_LINE(&HANDLE#,2)
:    SET_SCRIPT_VAR   &VARIABLE#=&WERT#
:    PRINT   "&VARIABLE# = &WERT#"
:ENDPROCESS

:CLOSE_PROCESS   &HANDLE#
```

Mit der PRINT-Anweisung werden die Werte des Variablen-Objektes im Aktivierungsprotokoll ausgegeben (Key = Wert 1).

Auszug aus dem Report:

```
2005-02-03 13:46:59 - U0020408 Ende = 20051027
2005-02-03 13:46:59 - U0020408 Start = 20051024
2005-02-03 13:46:59 - U0020408 Ultimo = 20051031
```

Siehe auch:

Script-Sprachmittel	Beschreibung
GET_SCRIPT_VAR	Liefert die Werte von Script-Variablen per indirektem Zugriff.
:PSET	Weist einer Objektvariablen einen Wert zu.

:RSET	Weist einer Script-Variablen einen Wert zu und speichert diesen im Aktivierungsreport.
:SET	Weist einer Script-Variablen einen Wert zu. Die Anweisung kann in Lang- oder Kurzform geschrieben werden.

Script-Sprachmittel - Script-Aufbau und -Prozessierung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5.20 :SWITCH ... :CASE ... :OTHER ... :ENDSWITCH

Script-Anweisung: Überprüft, ob der Wert einer Variable bestimmten Werten entspricht und führt abhängig davon unterschiedliche Anweisungen durch.

Syntax

:SWITCH *Variable*
:CASE *Ausdruck1*
[Anweisungen]
[:CASE *Ausdruck2*
[Anweisungen]
...
]
[:OTHER
[Anweisungen]
...
]
:ENDSWITCH

Syntaxteil	Beschreibung / Format
Variable	Name der Variable, deren Wert überprüft werden soll. Format: Script-Literal, Script-Variable
Ausdruck1, Ausdruck2 ...	Bedingungen oder Werte, welche die Bedingungen bilden. Format: Script-Literal, Script-Variable, komplexer Ausdruck oder Script-Funktion
Anweisungen	Ein oder mehrere Anweisungen, die durchgeführt werden sollen, wenn die Variable dem jeweiligen Wert entspricht. Format: Script-Anweisung

Anmerkungen

Die SWITCH-Anweisung bildet ein oder mehrere Bedingungen, indem sie abhängig vom Wert einer Variable unterschiedliche Anweisungen ausführt. Der :SWITCH-Block besteht aus ein oder mehreren :CASE-Anweisungen und ist mit :ENDSWITCH abzuschließen.

Mit jeder :CASE-Anweisung wird ein Wert oder ein Ausdruck festgelegt, der ausgewertet werden soll. Diesem folgen die Anweisungen, die im zutreffenden Fall auszuführen sind. Abgeschlossen werden diese Anweisungen entweder mit einer neuen :CASE-Zeile oder mit :ENDSWITCH.

Zusätzlich kann auch ein :OTHER Zweig definiert werden. Dessen Anweisungen werden dann ausgeführt, wenn keine einzige :CASE-Anweisung zutrifft. Die :OTHER Anweisung darf erst nach der letzten :CASE-Anweisung verwendet werden.

Mehrere :CASE Anweisungen hintereinander, ohne Anweisungen getrennt, sind ebenfalls möglich. Diese werden durch eine ODER-Verknüpfung miteinander verbunden. Dies macht allerdings nur bei komplexen Ausdrücken Sinn!

Mit :IF-Anweisungen können ebenfalls Ausdrücke ausgewertet werden, jedoch ist deren Anweisungs-Block auf eine Bedingung und eine Sonst-Bedingung beschränkt.

Für die *Werte* können Sie auch komplexe Ausdrücke, wie arithmetische Terme, verwenden. Beispiel:

```
:SWITCH &NUM#
:CASE 3*(5-2)
: PRINT "&NUM# = 9"
:ENDSWITCH
```

Auch das Konstrukt **between** Wert1 **and** Wert2, sowie die Operatoren <, >, <>, <=, >=, = können bei den :CASE-Zeilen verwendet werden. In diesem Fall ist anstelle des Variablennamens die Zeichenkette "Y" anzugeben. Beispiel:

```
:SET &STATUS# = GET_STATISTIC_DETAIL(&RUNID#,STATUS)

:SWITCH "Y"
:CASE &STATUS# between 1300 and 1599
:CASE &STATUS# between 1700 and 1799
: PRINT "Aufgabe &RUNID# ist aktiv"
:CASE &STATUS# between 1600 and 1699
: PRINT "Aufgabe &RUNID# befindet sich im Wartezustand"
:CASE &STATUS# between 1800 and 1899
: PRINT "Aufgabe &RUNID# abgebrochen"
:CASE &STATUS# >= 1900
: PRINT "Aufgabe &RUNID# wurde erfolgreich beendet"
:ENDSWITCH
```

Beispiel

Folgendes Beispiel ermittelt den aktuellen Wochentag als Zahl und überprüft diesen mittels SWITCH-Anweisung. Montags und Freitags wird ein bestimmter Job gestartet.

```
:SET &DATE# = SYS_DATE_PHYSICAL("TT.MM.JJJJ")
:SET &WEEKDAY# = WEEKDAY_NR("TT.MM.JJJJ:&DATE#")

:SWITCH &WEEKDAY#
:CASE 1
: SET &ACT# = ACTIVATE_UC_OBJECT(JOBS.MONDAY, WAIT)
:CASE 5
: SET &ACT# = ACTIVATE_UC_OBJECT(JOBS.FRIDAY, WAIT)
:OTHER
: PRINT "Keine Verarbeitung"
:ENDSWITCH
```

iehe auch:

Script-Sprachmittel	Beschreibung

:IF ... :ELSE ... :ENDIF	Werten einen Ausdruck aus und führen in Abhängigkeit vom Ergebnis Anweisungen aus.

Script-Sprachmittel - Datensequenzen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5.21 :WAIT

Script-Anweisung: Die Scriptbearbeitung wird für einen festgelegten Zeitraum unterbrochen.

Syntax

:WAIT*Zeit*

Syntaxteil	Beschreibung/Format
Zeit	Zeit in Sekunden, die gewartet werden soll. Format: Script-Variable oder Zahl

Anmerkungen

Die Script-Anweisung ermöglicht es, die Verarbeitung des Scripts für den mit *Zeit* festgelegten Zeitraum zu unterbrechen.

Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

Beispiel

Die Abarbeitung des Scriptes wird für 10 Sekunden unterbrochen.

```
:WAIT 10
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:STOP	Bricht die Bearbeitung eines Scripts ab.

Script-Sprachmittel - Script-Aufbau und -Prozessierung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5.22 :WHILE... :ENDWHILE

Script-Anweisungen: Definition einer Schleife zur mehrfachen Ausführung von Scriptteilen.

Syntax

:WHILE*Bedingung*
[Anweisungen]
:ENDWHILE

Syntaxteil	Beschreibung/Format
Bedingung	Ein mittels Script-Literalen, Script-Variablen, Script-Funktionen oder numerischen Ausdrücken sowie Operatoren formulierter Ausdruck, der als Ergebnis "Wahr" oder "Falsch" liefert.
Anweisungen	Eine oder mehrere Anweisungen, die ausgeführt werden, solange *Bedingung* den Wert "Wahr" besitzt. Format: Script-Anweisung

Anmerkungen

Die :WHILE-Anweisung ist eine Kontrollstruktur mit der Sie Teile eines Scriptes abhängig vom Ergebnis einer *Bedingung* mehrfach ausführen können. Bei dieser handelt es sich um einen logischen Ausdruck.

Ist die Bedingung erfüllt (Wahr), so werden die Anweisungen ausgeführt. Anschließend wird die Bedingung erneut ausgewertet und die Anweisungen nochmals ausgeführt, wenn sie "Wahr" ergibt. Dies wiederholt sich bis die Bedingung nicht mehr erfüllt ist (Falsch). In diesem Fall wird das Script mit der nächsten Zeile nach dem :ENDWHILE fortgesetzt.

⚠ Achten Sie darauf, dass eine Abbruchbedingung eintritt, da sonst eine Endlos-Schleife entsteht.

Sie können WHILE-Blöcke auch schachteln, wobei es keine Limitierung der Tiefe gibt.

Eine *Bedingung* weist folgende Struktur auf:

Wert ***Vergleichsoperator*** *Vergleichswert[*OR *Vergleichswert...]*

Beispiel:

```
:WHILE &ABT# = "EDV" OR "AUTOMIC"
```

Bis zu 13 OR dürfen in der Bedingung vorhanden sein.

Die Formulierung einer *Bedingung* besitzt eine der folgenden Vergleichsoperatoren:

Operator	Kürzel	Regel
=	EQ	Der Ausdruck ist "Wahr", wenn mindestens ein *Vergleichswert* dem *Wert* entspricht.
<>	NE	Der Ausdruck ist "Wahr", wenn keiner der *Vergleichswerte* gleich dem *Wert* ist.

< > <= >=	LT GT LE GE	Der Ausdruck ist "Wahr", wenn ein *Vergleichswert* den angegebenen Begriff erfüllt.

Da Script-Variablen keinen expliziten Datentyp haben und daher sowohl Zahlen als auch Zeichenfolgen beinhalten können, ist bei den Vergleichen Folgendes zu beachten:

Der Vergleich erfolgt immer dann numerisch, wenn die zu vergleichenden Werte inhaltlich numerisch sind. Dabei spielt es keine Rolle, ob die Variable mit oder ohne Hochkomma angegeben wurde.

Die folgende Bedingung liefert "Wahr":

```
:WHILE "0" = "0"
```

Handelt es sich um keine numerischen Werte, so erfolgt ein alphanumerischer Vergleich.

Die folgende Bedingung liefert "Falsch":

```
:WHILE "0" = "abc"
```

Beim Vergleich von Zeichenfolgen wird nicht deren Länge geprüft. Die folgende Bedingung liefert "Falsch", da "A" im Alphabet vor "M" steht.

```
:WHILE "Meier" < "AUTOMIC"
```

Beispiel

Das Beispiel erfragt vom Benutzer einen Namen und eine Abteilung. Die Abfrage wird solange wiederholt, bis der Benutzer einen Namen eingegeben hat, der nicht sein eigener ist.

```
:SET &WIEDERHOLEN# = "J"

:WHILE &WIEDERHOLEN# = "J"
:SET &ABT# = SYS_USER_DEP()
:SET &NAME# = SYS_USER_NAME()

:BEGINREAD
:   READ &ABT#, "08" ,"Bitte Abteilung eingeben", &ABT#, "M"
:   READ &NAME#, "08", "Bitte Namen eingeben", &NAME#, "M"
:ENDREAD

:IF SYS_LAST_ERR_NR() <> 0
:   STOP
:ELSE
:   IF SYS_USER_NAME() = &NAME#
:      SET &WIEDERHOLEN# = "J"
:      SET_LAST_ERR 10006, "Der Name darf nicht der eigene sein"
:   ELSE
:      SET &WIEDERHOLEN# = "N"
:   ENDIF
:ENDIF

:ENDWHILE
```

Siehe auch:

Script-Sprachmittel	Beschreibung

:IF... :ELSE... :ENDIF	Werten einen Ausdruck aus und führen in Abhängigkeit vom Auswertungsergebnis Anweisungen aus.

Script-Sprachmittel - Script-Aufbau und -Prozessierung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5.23 FIND

Script-Funktion: Durchsucht ein Script-Array und liefert den entsprechenden Index zurück.

Syntax

FIND*(Script-Array, Suchbegriff *[, Startposition]**)***

Syntaxteil	Beschreibung/Format
Script-Array	Name des zu durchsuchenden Script-Arrays. Format: Script-Variable
Suchbegriff	Zeichenkette oder Zahl nach der gesucht werden soll. Format: Format: String-Literal, Zahl ohne Hochkommata, Script-Variable
Startposition	Index, ab dem die Suche gestartet werden soll. Format: Format: Zahl ohne Hochkommata, Script-Variable
Rückgabewerte	
Index des Arrays, in dem der Suchbegriff gefunden wurde. 0 = Kein Suchergebnis	

Anmerkungen

Die Suche nach *Suchbegriff* wird für alle Werte von *Script-Array* ab dem Index *Startposition* durchgeführt. Wird keine Startposition angegeben, wird das ganze Array durchsucht (ab Index 1). War die Suche erfolgreich, wird der Array-Index zurückgeliefert, in dem der Text bzw. die Zahl als erstes gefunden wurde. Um die Suche im restlichen Teil des Array fortzusetzen, ist als *Startposition* der nächste Index des vorigen Ergebnisses zu verwenden. Liefert die Suche keinen Treffer, so erhält man 0 als Rückgabewert der Script-Funktion.

Das Script-Array ist mit leeren Indexklammern [] anzugeben.

Script-Arrays müssen mit :DEFINE erstellt werden, wobei auch Datentyp und Größe festgelegt werden. Felder des Arrays, denen noch kein bestimmter Wert zugewiesen wurde, besitzen den Standardwert "" (Datentyp: string) oder 0 (numerische Datentypen). Die Suche mit FIND ist daher nur sinnvoll, wenn das Array auch entsprechend befüllt wurde. Die Array-Elemente können einzeln mit Angabe des Index mit :SET gesetzt werden. Um mehrere verschiedene Werte gleichzeitig in ein Array abzuspeichern, steht die Script-Anweisung :FILL zur Verfügung.

Beispiele

Das Beispiel zeigt die Erstellung und Initalisierung eines Arrays mit Werten aus einem Variablen-Objekt. Anschließend wird das gesamte Array nach WIN durchsucht und alle Ergebnisse in das Aktivierungsprotokoll geschrieben.

```
:DEFINE &array#, string, 5
:FILL &array#[] = GET_VAR(VARA.WIN, AGENTS)
:SET &search# = FIND(&array#[],"WIN", 1)
:WHILE &search# <> 0
 :PRINT "WIN gefunden bei Position: &search#"
 :SET &search# = &search# + 1
 :SET &search# = FIND(&array#[], "WIN", &search#)
:ENDWHILE
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:FILL	Speichert mehrere Werte in ein Script-Array.
LENGTH	Ermittelt die Größe eines Script-Arrays.

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5.24 GET_SCRIPT_VAR

Script-Funktion: Liefert die Werte von Script-Variablen per indirektem Zugriff.

Syntax

GET_SCRIPT_VAR*(Script-Variable)*

Syntaxteil	Beschreibung/Format
Script-Variable	Platzhalter für die Namen der Script-Variablen, deren Werte ausgelesen werden sollen. Format: AE-Name, Script-Literal oder Script-Variable

Anmerkungen

Die Script-Funktion liest die Werte von Script-Variablen, ohne dass dabei die Namen der Script-Variablen explizit angegeben werden müssen. Auf die Script-Variablen wird indirekt über einen Platzhalter, man könnte ihn selbst eine Variable nennen, zugegriffen. Dadurch ist es möglich, komfortabel die Werte vieler Script-Variablen, beispielsweise in einer Prozess-Schleife, abzufragen. Die Script-Funktion ersetzt so viele bisher notwendige Bedingungsanweisungen durch eine einzige Script-Zeile.

Der Script-Funktion wird mit *Script-Variable* eine Zeichenkette übergeben, aus welcher der Name einer Script-Variable gebildet wird. Dabei ist es nicht notwendig, dass die Zeichenkette bereits mit

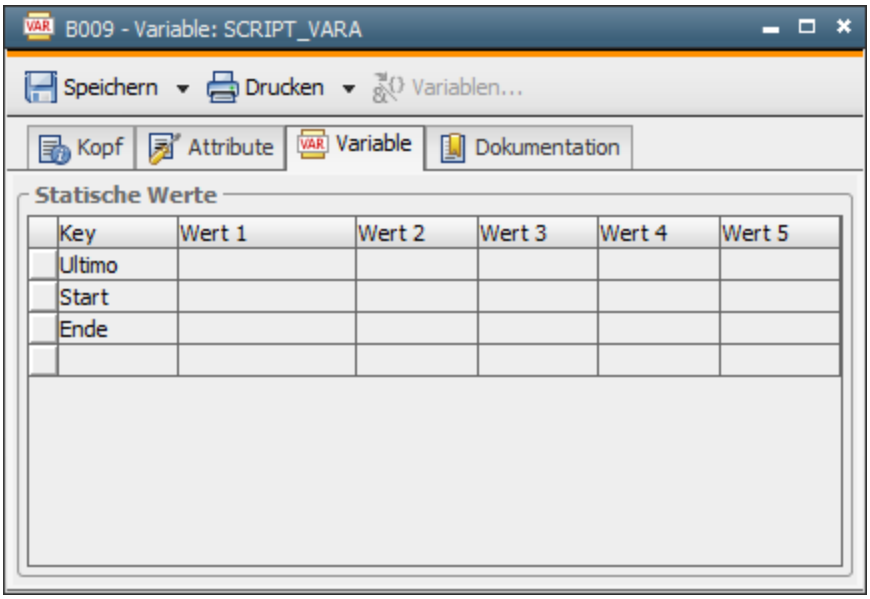
einem "&" beginnt, wodurch eine Script-Variable gekennzeichnet ist. Es sind allerdings mindestens soviele Anfangsbuchstaben des Variablennamens anzugeben, die erforderlich sind, um die Script-Variable eindeutig zu identifizieren. Ist die Angabe nicht eindeutig, so wird der Wert der ersten zutreffenden Variable gelesen.

Der Variablenname wird ab & oder dem Zeichen danach gesucht. Stimmt die angegebene Zeichenkette mit dem Beginn keines Variablennamens überein, so kommt es zu einem Fehler.

Mit dem Sprachmittel ist es auch möglich, die Werte von Objekt- und PromptSet-Variablen zu ermitteln.

Beispiel

Das Beispiel basiert auf einem Variablen-Objekt und einem Job. In dem Variablen-Objekt sind die Namen von Script-Variablen gespeichert.



Im Script des Jobs wird in einer Prozess-Schleife zunächst der Name der Script-Variablen aus dem Variablen-Objekt gelesen. Dieser wird der Variable "&WERT#" übergeben. Der Name der Script-Variablen und ihr Wert werden im Aktivierungsreport ausgegeben. Dies kann in einer PRINT-Anweisung mit der Script-Variablen "&WERT#" geschehen, welche nacheinander den Inhalt von "&ENDE#", "&START#" und "&ULTIMO#" enthält.

```
:SET   &ENDE#    = "20051027"
:SET   &START#   = "20051024"
:SET   &ULTIMO# = "20051031"

:SET   &HANDLE# = PREP_PROCESS_VAR("SCRIPT_VARA")

:PROCESS   &HANDLE#
:    SET   &VARIABLE# = GET_PROCESS_LINE(&HANDLE#,1)
:    SET   &WERT# = GET_SCRIPT_VAR(&VARIABLE#)
:    PRINT   "&VARIABLE# = &WERT#"
:ENDPROCESS

:CLOSE_PROCESS   &HANDLE#
```

Auszug aus dem Report:

```
2005-02-03 12:51:23 - U0020408 Ende = 20051027
2005-02-03 12:51:23 - U0020408 Start = 20051024
2005-02-03 12:51:23 - U0020408 Ultimo = 20051031
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:SET_SCRIPT_VAR	Setzt die Werte von Script-Variablen per indirektem Zugriff.
:RSET	Weist einer Script-Variablen einen Wert zu und speichert diesen im Aktivierungsreport.

Script-Sprachmittel - Script-Aufbau und -Prozessierung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.5.25 LENGTH

Script-Funktion: Ermittelt die Größe eines Script-Arrays.

Syntax

LENGTH*(Script-Array [,* **SIZE***])*

Syntaxteil	Beschreibung/Format
Script-Array	Name der Script-Variable, die als Array deklariert wurde. Format: Script-Variable
SIZE	Leere Elemente am Ende des Arrays ignorieren
Rückgabewerte	
Anzahl der Elemente des Script-Arrays.	

Anmerkungen

Script-Arrays müssen mit dem Sprachmittel :DEFINE erstellt werden, wobei auch Datentyp und Anzahl der Elemente festgelegt wird. Danach kann die Größe des Arrays mit der Script-Funktion LENGTH abgefragt werden. Das Array ist dabei mit leeren Indexklammern anzugeben.

Die maximale Größe von Script-Arrays beträgt 99999.

Wurde das angegebene Array nicht gefunden oder eine Script-Variable angegeben, kommt es zu einem Scriptfehler.

Wird zusätzlich der Parameter SIZE spezifiziert, so wird nur die Anzahl bis zum letzten befüllten Array-Element zurückgeliefert. Alle leeren Elemente am Ende des Arrays werden somit nicht mitgezählt.

⚠ Beachten Sie, dass die Ermittlung der Größe bis zum letzten befüllten Element (Parameter SIZE) nur möglich ist, wenn das Array den Datentyp "string" aufweist. Bei anderen Datentypen wird immer die gesamte Array-Größe zurückgeliefert!

Beispiele

Im folgenden Beispiel wird ein Script-Array erstellt, mit Werten aus einem Variablen-Objekt befüllt und danach die Werte mit Hilfe einer Schleife ausgegeben.

```
:DEFINE &ARRAY#, string, 5
:SET &LEN# = LENGTH(&ARRAY#[])
:SET &VAR# = 1
:FILL &ARRAY#[] = GET_VAR(VARA1, ARRAY)
:WHILE &VAR# LE &LEN#
 :PRINT "Element &VAR# = &ARRAY#[&VAR#]"
 :SET &VAR# = &VAR# + 1
:ENDWHILE
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:FILL	Speichert mehrere Werte in ein Script-Array.
FIND	Durchsucht ein Script-Array und liefert den entsprechenden Index zurück.

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.6 Fehlerbehandlung und Meldungen

3.6.1 :EXIT

Script-Anweisung: Beendet die Prozessierung des Scripts mit Rückgabewert.

Syntax

:EXIT *[Rückgabewert]*

Syntaxteil	Beschreibung/Format
Rückgabewert	Wert, welcher als Rückgabewert gesetzt werden soll. Format: Zahl, Script-Variable oder Script-Funktion Standardwert: "0"

Anmerkungen

Die Script-Anweisung ermöglicht es, die Prozessierung eines Scripts mit einem angegebenen *Rückgabewert* zu beenden. Sie wird verwendet, um dabei auf Bedingungen zu reagieren, die zum Abbruch der Aufgabe führen. Beispielsweise könnte so in einem Script-Objekt auf den Inhalt einer Variablen geprüft und in Abhängigkeit davon, die Aufgabe fortgesetzt oder abgebrochen werden.

Wird die Script-Anweisung ohne *Rückgabewert* oder mit dem Wert "0" verwendet, endet die Prozessierung des Scripts an dieser Stelle normal mit Rückgabewert "0". Die Aufgabe wird aber dennoch fortgesetzt, d.h. Jobs werden mit der generierten JCL gestartet, Benachrichtigungen werden angezeigt usw.

Als *Rückgabewert* können auch Benutzer-Rückgabewerte angegeben werden. Die Prozessierung des Scripts wird abgebrochen und die Aufgabe abnormal beendet (ENDED_NOT_OK). Ist in der Registerkarte **Attribute** die Aufgabe die Generierung zur Laufzeit aktiviert, kann auf diesen Benutzer-Rückgabewert innerhalb eines Workflows reagiert werden.

⚠ :EXIT beendet die Prozessierung des Scripts. Deshalb kann das Sprachmittel nicht in Post-Script-Registerkarten verwendet werden. Nutzen Sie in solchen Fällen die Anweisung :MODIFY_STATE.

Beispiel

Das erste Beispiel beendet die Script-Prozessierung mit dem Rückgabewert 10.

```
:EXIT 10
```

Siehe auch:

:MODIFY_STATE	Ändert den Rückgabewert oder Statustext eines Jobs nach dessen Ende.
:STOP	Bricht die Aktivierung eines Scripts ab.

Script-Sprachmittel - Fehlerbehandlung und Meldungen
System-Rückgabewerte von aktivierbaren Objekten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.6.2 :ON_ERROR

Script-Anweisung: Legt die Reaktion auf bestimmte Fehler bzw. Meldungen von Script-Sprachmitteln fest.

Syntax

:ON_ERROR*Reaktion*

Syntaxteil	Beschreibung/Format

Reaktion	Schlüsselwort für die Reaktion auf Fehler bzw. Meldungen. Format: AE-name, Script-Literal oder Script-Variable Erlaubte Werte: "ABEND", "RESUME" (Standardwert) "ABEND" - Script-Prozessierung wird abgebrochen "RESUME" - Script-Prozessierung wird fortgesetzt

Anmerkungen

Die Script-Anweisung beeinflusst folgende Script-Funktionen:

ACTIVATE_UC_OBJECT	LAST_OF_PERIOD
AUTOFORECAST	MODIFY_OBJECT
CANCEL_UC_OBJECT	MOVE_OBJECT
CHANGE_LOGGING	PREP_PROCESS_FILENAME
CREATE_OBJECT	REMOVE_OBJECT
EXPORT	RESTART_UC_OBJECT
FIRST_OF_PERIOD	SEND_MAIL
GET_FILESYSTEM	SEND_MSG
GET_OBJECT_TYPE	SYS_SERVER_ALIVE
GET_STATISTIC_DETAIL	SYS_USER_ALIVE
IMPORT	

AE arbeitet standardmäßig so, dass bei diesen Script-Funktionen die Script-Prozessierung nicht abbricht, wenn bestimmte Fehler auftreten. Der Fehler kann nach Auftreten mit den Script-Funktionen zur Fehlerbehandlung analysiert werden.

Mit dem Parameter "ABEND" zwingen Sie das Script bei Fehlern zum Abbruch. Bei der Verwendung von "RESUME" hingegen, wird das Script fortgesetzt und die Rückgabewerte der zuvor aufgelisteten Funktionen sind auslesbar.

Beispiel

Im Beispiel wird festgelegt, dass bei einem Fehler das Script nicht abbricht. Anschließend wird versucht, die Speicherkapazität eines nicht vorhandenen Laufwerkes zu prüfen. Dabei kommt es zu einem Fehler, der mit den Script-Funktionen zur Fehlerbehandlung analysiert wird.

```
:ON_ERROR RESUME
:SET &CHECK# = GET_FILESYSTEM('WIN21', 'Z:\', FILESYSTEM_SPACE_TOTAL)
:SET &ERRNR# = SYS_LAST_ERR_NR
:SET &ERRINS# = SYS_LAST_ERR_INS
:SET &MESSAGE# = GET_MSG_TXT (&ERRNR#,&ERRINS#)
```

Siehe auch:

Script-Sprachmittel - Fehlerbehandlung und Meldungen
Beispielsammlung
Ermittlung der Fehlermeldung und -nummer

Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.6.3 :SEND_MSG

Script-Anweisung: Schickt Nachrichten an Benutzer, die am UserInterface angemeldet sind.

Syntax

:SEND_MSG *Name, Abteilung, Nachricht*

Syntaxteil	Beschreibung/Format
Name	Name des Benutzers, der die Nachricht erhalten soll. Format: Script-Literal, Script-Variable oder Script-Funktion Die Wildcardzeichen "*" und "?" können verwendet werden. "*" steht dabei für beliebige, "?" für genau ein Zeichen.
Abteilung	Abteilung des Benutzers, der die Nachricht erhalten soll. Format: Script-Literal, Script-Variable oder Script-Funktion Die Wildcardzeichen "*" und "?" können verwendet werden. "*" steht dabei für beliebige, "?" für genau ein Zeichen.
Nachricht	Nachricht, welche verschickt werden soll. Format: Script-Literal, Script-Variable oder Script-Funktion

Anmerkungen

Mit dieser Script-Anweisung können Sie Benutzern Nachrichten zukommen lassen. Diese werden in deren Meldungsfenster angezeigt.

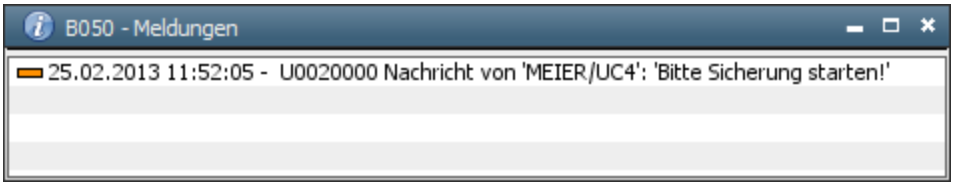
Da die Nachricht nur an Benutzer zugestellt wird, die am UserInterface angemeldet sind, empfielt sich die Verwendung des Script-Sprachmittels SYS_USER_ALIVE, das diesen Umstand überprüft.

Existiert der Benutzer nicht, so tritt der Fehler "20698" auf. Mit der Script-Anweisung :ON_ERROR legen Sie die Reaktion darauf fest. Wie bisher können Sie ihn mit den Script-Funktionen zur Fehlerbehandlung analysieren. Das Script wird weiter prozessiert. Es besteht aber auch die Möglichkeit, die Prozessierung des Scripts abzubrechen.

Die versendeten Nachrichten werden mitprotokolliert und sind in der Kategorie "Meldungen" der Systemsteuerung zu finden. Der Meldungstyp ist "Information" und die Meldungskategorie lautet "Nachricht".

⚠ Beachten Sie, dass die gesendeten Nachrichten nur dann angezeigt werden, wenn die UserInterface - Option "Nur Fehlermeldungen und Warnungen anzeigen" nicht aktiviert ist.

Mit dem Privileg "Alle Meldungen seines Mandanten sehen" werden auch die Nachrichten angezeigt, die an andere Benutzer gesendet werden.



Beispiele

Der Benutzer, der mit dem Namen "HUBER" und der Abteilung "EDV" am UserInterface angemeldet ist, erhält eine Aufforderung zur Sicherung. Sollte jener nicht am UserInterface angemeldet sein, wird ein Benachrichtigungs-Objekt gestartet.

```
:SET &RET# = SYS_USER_ALIVE("MEIER","EDV")

:IF &RET# = "Y"
:  SEND_MSG "HUBER","EDV","Bitte Sicherung starten!"
:ELSE
:  SET &ACTOBJ# = ACTIVATE_UC_OBJECT(CALL,"TAGSCHICHT")
:ENDIF
```

Der Benutzer erhält die gleiche Nachricht. Dabei wird für die *Abteilung* eine Script-Funktion und für die *Nachricht* eine Script-Variable verwendet.

```
:SET &MSG# = "Bitte Sicherung starten!"
:SEND_MSG "HUBER",SYS_USER_DEP(),&MSG#
```

Dieses Beispiel verwendet Wildcard-Zeichen, um die Nachricht an alle Benutzer der Abteilung "EDV" zu versenden.

```
:SEND_MSG "*","EDV","Bitte Sicherung starten!"
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_USER_ALIVE	Prüft, ob der Benutzer mit einem UserInterface an der Automation Engine angemeldet ist.
SEND_MAIL	Sendet E-Mail an einen Benutzer.

Script-Sprachmittel - Fehlerbehandlung und Meldungen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.6.4 :SEND_SNMP_TRAP

Script-Anweisung: Sendet einen SNMP-Trap.

Syntax

:SEND_SNMP_TRAP ***Trap-Code, Parameter*** *[, Parameter[, Parameter]]...*

Syntaxteil	Beschreibung/Format
Trap-Code	Frei definierbarer Code, auf den ein SNMP-Management reagieren kann. Format: Zahl, Script-Variable oder Script-Funktion
Parameter	Zusatzinformationen, die mit dem SNMP-Trap geschickt werden können. Format: Zahl, Script-Literal, Script-Variable oder Script-Funktion Die Mandantennummer ist unter Hochkomma anzugeben (z.B. "1000").

Anmerkungen

Die Script-Anweisung sendet einen SNMP-Trap mit den angegebenen Parametern an ein Management-System. Sie kann nur eingesetzt werden, wenn zuvor alle Voraussetzungen für die SNMP-Funktionalität realisiert wurden.

Es können bis zu 10 *Parameter* mitgeschickt werden. Davon dürfen maximal 5 Zeichenketten und maximal 5 Zahlen sein. Die Position von Zeichenketten und Zahlen ist fest vorgegeben. Zuerst müssen die Zeichenketten, dahinter die Zahlen stehen. Es muss mindestens eine Zeichenkette angegeben werden.

⚠ Von AE und von HP OpenView Integrator (Trap-Codes: 10000 - 10010) generierte Trap-Codes dürfen nicht verwendet werden.

Beispiele

Das erste Beispiel zeigt einen einfachen Test. Mit dem SNMP-Trap 50000 werden alle möglichen Parameter an ein Management-System geschickt.

```
:SEND_SNMP_TRAP
50000,"Text1","TEXT2","TEXT3","TEXT4","TEXT5",1,2,3,4,5
```

Im zweiten Beispiel werden zunächst die eigenen Aktivierungsdaten ermittelt. Abhängig davon, ob es sich um eine direkte Aktivierung oder um eine Aktivierung innerhalb eines Workflows handelt, werden unterschiedliche Traps geschickt. Beide Traps bestehen aus dem Trap-Code und einer aufbereiteten Zeichenkette.

```
:SET &NAME#      = SYS_ACT_ME_NAME()
:SET &ID#        = SYS_ACT_ME_NR()
:SET &JPNAME#    = SYS_ACT_PARENT_NAME()
:SET &CLIENT#    = SYS_ACT_CLIENT()
!Trap-Typ = Alarm
:SET &TYPE#      = 4
!Wichtigkeitsgrad des Ereignisses
:SET &SEV#       = 4


:IF  "&JPNAME#" = ""
:    SEND_SNMP_TRAP 50001, "&CLIENT#","Fehler in der Aufgabe &NAME#
(&ID#)!",,,,&TYPE#,&SEV#
:ELSE
:    SET &JPID# = SYS_ACT_PARENT_NR()
:    SEND_SNMP_TRAP 50002, "&CLIENT#","Fehler im Workflow &JPNAME#
(&JPID#) bei der Aufgabe &NAME# (&ID#)!",,,,&TYPE#,&SEV#
:ENDIF
```

Das dritte Beispiel zeigt, wie die ermittelten Aktivierungsdaten der Script-Anweisung als einzelne Parameter übergeben werden. Damit stehen die Aktivierungsdaten im Management-System auch als einzelne Werte zur Verfügung und können dort einfacher weiterverarbeitet werden.

```
:SET &NAME#      = SYS_ACT_ME_NAME()
:SET &ID#        = SYS_ACT_ME_NR()
:SET &JPNAME#    = SYS_ACT_PARENT_NAME()
:SET &CLIENT#    = SYS_ACT_CLIENT()
!Trap-Typ = Alarm
:SET &TYPE#      = 4
!Wichtigkeitsgrad des Ereignisses
:SET &SEV#       = 4



:IF   &JPNAME# = ""
:    SEND_SNMP_TRAP 50001,"&CLIENT#","&NAME#",,"Fehler bei der
Aufgabe!",,&TYPE#,&SEV#,&ID#
:ELSE
:    SET &JPID = SYS_ACT_PARENT_NR()
:    SEND_SNMP_TRAP 50002,"&CLIENT#","&NAME#","&JPNAME#","Fehler im
Workflow!",,&TYPE#,&SEV#,&ID#,&JPID#
:ENDIF
```

Sollen &ID# und/oder &JPID# nicht als Zeichenketten sondern als Zahlen gesendet werden, dürfen sie erst ab dem sechsten Parameter übergeben werden. Anstelle der ausgelassenen Parameter, die für Zeichenketten vorgesehen sind, müssen die Kommas stehen.

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_SNMP_ACTIVE	Prüft, ob die SNMP-Kopplung (Simple Network Management Protocol) von AE aktiv ist.

Script-Sprachmittel - Fehlerbehandlung und Meldungen
AE und SNMP
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.6.5 :SET_LAST_ERR

Script-Anweisung: Setzt Fehlernummer und -text.

Syntax

:SET_LAST_ERR*Nummer*[, *Insert*]

Syntaxteil	Beschreibung/Format
Nummer	Fehlernummer. Format: Zahl, Script-Variable oder Script-Funktion

Insert	Fehlertext, variabler Teil der Fehlermeldung. Format: Script-Literal, Script-Variable oder Script-Funktion

Anmerkungen

Die Script-Anweisung :SET_LAST_ERR setzt Fehlernummer und -text. Die Anweisung überschreibt dabei Fehlernummer und/oder variablen Teil der letzten Fehlermeldung. Die Angabe von *Insert* ist optional. Wird *Insert* nicht angegeben, ist der variable Teil der Fehlermeldung ein Leerstring.

Achten Sie darauf bei der Fehlermeldung eine gültige anzugeben. Eine Liste aller verfügbaren Meldungen mit ihren Texten finden Sie im Meldungshandbuch.

Besitzt die Meldung mehrere variable Teile, so sind diese mit einem "|" getrennt anzugeben. Die Reihenfolge ist ebenfalls im Meldungshandbuch ersichtlich.

Mit :SET_LAST_ERR 0 können Fehlernummer und -text gelöscht werden.

Die gesamte geänderte Fehlermeldung erhalten Sie mit GET_MSG_TXT.

Beispiel

Im Beispiel wird die Fehlernummer "20657" zusammen mit den beiden variablen Meldungsteilen gesetzt.

```
:SET_LAST_ERR 20657,"MAWI.TAG|20||OBJEKTE"
 :SET &ERRINS# = SYS_LAST_ERR_INS()
 :SET &ERRNR# = SYS_LAST_ERR_NR()
 :SET &MSG# = GET_MSG_TXT(&ERRNR#, &ERRINS#)
 :PRINT &MSG#
```

Auszug aus dem Meldungshandbuch: 20657 - Laufzeitfehler in Objekt '&01' Zeile '&02': Zielordner '&04' nicht gefunden.

Das Ergebnis der Script-Anweisung lautet:

```
U0020657 Laufzeitfehler in Objekt 'MAWI.TAG' Zeile '20': Zielordner
'OBJEKTE' nicht gefunden.
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_LAST_ERR_INS	Liefert den variablen Teil der Fehlermeldung bei einem zuletzt aufgetretenen Fehler.
SYS_LAST_ERR_NR	Gibt die Fehlernummer eines zuletzt aufgetretenen Fehlers zurück.
SYS_LAST_ERR_SYSTXT	Ermittelt den Fehlertext des Betriebssystems eines zuletzt aufgetretenen Fehlers.

Script-Sprachmittel - Fehlerbehandlung und Meldungen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.6.6 :STOP

Script-Anweisung: Bricht die Aktivierung eines Scripts ab.

Syntax

:STOP *[Stopmodus]*

Syntaxteil	Beschreibung/Format
Stopmodus	Der :STOP-Anweisung können die folgenden Ausdrücke als *Stopmodus* übergeben werden: Keine Angabe Bricht die Bearbeitung eines Scripts ab und gibt das Aktivierungsprotokoll mit der Meldung U0010014 aus. NOMSG*, Meldungsnummer[, Meldungstext]* Beendet die Aktivierung des Objektes ohne Fehler. Es kann eine Meldung erzeugt werden. MSG, *Fehlernummer*, *Fehlertext* Bricht die Aktivierung eines Objektes mit einer Fehlermeldung ab.

Anmerkungen

⚠ :STOP bricht die Aktivierung eines Objektes ab. Deshalb kann das Sprachmittel nicht in Post-Script-Registerkarten verwendet werden. Nutzen Sie in solchen Fällen die Anweisung :MODIFY_STATE.

Stopmodus NOMSG

Die :STOP-Anweisung mit dem *Stopmodus* NOMSG kann in Objekten verwendet werden, die keine Ausführung durch einen Agenten erfordern. Dadurch ist es möglich, ausschließlich mit Script-Sprachmitteln zu arbeiten, also beispielsweise Variablen zu setzen.

Beim *Stopmodus* NOMSG kann zusätzlich eine Meldung erzeugt werden. Die *Meldungsnummer* ist syntaktisch notwendig, wird aber nicht ausgewertet. Der *Meldungstext* wird in der Statistik gespeichert und bei einem Aufruf über das CallAPI als Teil der Quittung geschickt. Damit können Sie dem CallAPI beispielsweise die laufende Nummer (RunID) einer gestarteten Aufgabe zurückgeben.

Da *Stopmodus* NOMSG die Aktivierung einer Aufgabe ohne Fehler beendet, wird diese in der Statistik nicht als abgebrochene Aufgabe angezeigt.

Stopmodus MSG

Den *Stopmodus* MSG können Sie benutzen, um die Aktivierung eines Objektes mit einer Fehlermeldung abzubrechen. Es kann dabei eine *Fehlernummer* und ein *Fehlertext* angegeben werden. Das ist besonders wichtig für Objekte, welche über das CallAPI von der Automation Engine gestartet werden. Die so zurückgegebenen Informationen bilden dort die einzige Möglichkeit, etwas über die Durchführung des Scripts zu erfahren. Die Fehlernummern 50 bis 59 sind für Benutzer reserviert.

Die Fehlernummer 50 führt zum Rückgabewert 4 und die Fehlernummern 51 bis 59 beenden das Dienstprogramm mit Rückgabewert 8.

Beachten Sie, dass nur die Fehlernummern 50 - 59 verwendet werden können! Andere Fehlernummern führen zum Abbruch des Scripts mit der Meldung, dass die angegebene Fehlernummer ungültig ist.

Da *Stopmodus* MSG die Aktivierung einer Aufgabe mit Fehler beendet, wird diese in der Statistik als abgebrochene Aufgabe angezeigt (FAULT_OTHER - Start nicht möglich).

Wird die :STOP-Anweisung ohne Parameter bzw. mit dem *Stopmodus* MSG verwendet, bricht die Script-Verarbeitung mit Fehler ab. Dabei findet ein Datenbank-Rollback statt. Alle noch nicht ausgeführten Transaktionen werden rückgängig gemacht. Haben Sie im Script Variablen-Objekte versorgt, kann es dadurch möglich sein, dass die zu Beginn des Scripts stehenden Variablen-Objekte ihre neuen Werte bereits gespeichert haben, die zum Ende des Scripts versorgten jedoch nicht.

Das System schreibt Transaktionen in der AE Datenbank fest (Commit), wenn die Script-Verarbeitung unterbrochen wird. Dieser Zeitpunkt tritt

- automatisch in Abständen von 5 Sekunden,
- durch Script-Sprachmittel, wie beispielsweise ACTIVATE_UC_OBJECT, GET_FILESYSTEM, :READ, PREP_PROCESS, PREP_PROCESS_FILE,
- und durch die Script-Anweisung :WAIT ein.

Mit der Anweisung :WAIT 0 erzwingen Sie eine Unterbrechung der Script-Verarbeitung und somit ein Datenbank-Commit.

Beispiel

Das Beispiel gibt eine Fehlermeldung in Abhängigkeit davon aus, ob ein Job korrekt aktiviert wurde oder nicht.

```
:SET &AKTNR# = ACTIVATE_UC_OBJECT(JOBS, MAWI)
:IF &AKTNR# = "0"
: STOP MSG, 50, "Fehler bei der Aktivierung des Jobs MAWI."
:ELSE
: PRINT "Der Job wurde mit der Aktivierungsnummer &AKTNR# gestartet."
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:EXIT	Beendet die Prozessierung des Scripts mit Rückgabewert.
:WAIT	Die Scriptbearbeitung wird für einen festgelegten Zeitraum unterbrochen.

Script-Sprachmittel - Objekte aktivieren
Script-Sprachmittel - Fehlerbehandlung und Meldungen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.6.7 GET_MSG_TXT

Script-Funktion: Ermittelt den Meldungstext des zuletzt aufgetretenen Fehlers.

Syntax

GET_MSG_TXT*(Nummer *[, Insert])***

Syntaxteil	Beschreibung/Format
Nummer	Fehlernummer Format: Script-Variable oder Zahl Die Fehlernummer wird auch von der Script-Funktion SYS_LAST_ERR_NR geliefert.
Insert	Variabler Teil der Fehlermeldung Format: Script-Literal oder Script-Variable Der variable Teil der Fehlermeldung wird auch von der Script-Funktion SYS_LAST_ERR_INS geliefert.

Rückgabewerte
Meldungstext des zuletzt aufgetretenen Fehlers. " " - Es ist kein Fehler aufgetreten.

Anmerkungen

Bei einigen ausgewählten Script-Sprachmitteln kann durch der Verwendung von :ON_ERROR das Script bei Fehlern trotzdem fortgesetzt werden. In diesen Fällen ist die Meldung des Fehlers mit der Funktion ermittelbar.

Grundlage der Ermittlung des Meldungstexts ist die Fehlernummer *Nummer* und der variable Teil der Fehlermeldung *Insert*. Die Fehlernummer kann zuvor mit SYS_LAST_ERR_NR, der variable Teil der Fehlermeldung mit SYS_LAST_ERR_INS ermittelt werden. Die Script-Funktion GET_MSG_TXT setzt mit diesen Angaben und den konstanten Fehlertexten die komplette Fehlermeldung zusammen.

Der Parameter *Insert* ist dann nicht anzugeben, wenn der Meldungstext keine variablen Teile enthält. Dies trifft beispielsweise auf Statustexte zu deren Nummer Sie mit GET_STATISTIC_DETAIL auslesen können.

Beispiel

Das Beispiel versucht die Speicherkapazität eines nicht vorhandenen Laufwerkes zu prüfen. Dabei kommt es zu einem Fehler. Fehlernummer und variabler Teil der Fehlermeldung werden gelesen. Mit diesen Informationen wird die komplette Fehlermeldung ermittelt und als Nachricht an einen Benutzer verschickt.

```
:SET &CHECK# = GET_FILESYSTEM("WIN21", "Z:\", FILESYSTEM_SPACE_TOTAL)
:SET &ERRNR# = SYS_LAST_ERR_NR()
:SET &ERRINS# = SYS_LAST_ERR_INS()
```

```
:SET &MESSAGE# = GET_MSG_TXT(&ERRNR#,&ERRINS#)
:SEND_MSG BU,BU,&MESSAGE#
```

Im zweiten Beispiel wird der Text zum Status einer Aufgabe ermittelt.

```
:SET &STATUS# = GET_STATISTIC_DETAIL(&RUNID#, STATUS)
:SET &TEXT# = GET_MSG_TXT(&STATUS#)
:PRINT &TEXT#
```

Die Ausgabe könnte wie folgt aussehen:

```
ENDED_CANCEL - beendet durch manuellen Abbruch
```

Siehe auch:

Script-Sprachmittel	Beschreibung
GET_MSG_TYPE	Ermittelt den Typ einer Meldungsnummer.
SYS_LAST_ERR_INS	Liefert den variablen Teil der Fehlermeldung bei einem zuletzt aufgetretenen Fehler.
SYS_LAST_ERR_NR	Gibt die Fehlernummer eines zuletzt aufgetretenen Fehlers zurück.

Script-Sprachmittel - Fehlerbehandlung und Meldungen
Beispielsammlung
Ermittlung der Fehlermeldung und -nummer
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.6.8 GET_MSG_TYPE

Script-Funktion: Ermittelt den Typ einer Meldungsnummer.

Syntax

GET_MSG_TYPE*(Meldungsnummer)*

Syntaxteil	Beschreibung/Format
Meldungsnummer	Nummer einer Meldung, deren Typ ausgelesen werden soll. Format: Script-Variable oder Zahl

Rückgabewerte
"A" - Abbruch "E" - Fehler "I" - Information "Q" - Frage "W" - Warnung " " - Der Fehlernummer existiert nicht.

Anmerkungen

Meldungsnummern werden in unterschiedliche Typen eingeteilt zu denen beispielsweise Warnungen und Fehlermeldungen gehören.

Mit der Script-Funktion SYS_LAST_ERR_NR erhalten Sie die Nummer der zuletzt aufgetretenen Meldung.

Beispiel

Im folgenden Beispiel wird der Administrator verständigt, wenn eine Fehler- oder Abbruchsmeldung eintritt.

```
:SET &ERRNR# = SYS_LAST_ERR_NR()
:SET &ERRTYP# = GET_MSG_TYPE(&ERRNR#)
:IF &ERRTYP# = "A" OR "E"
:   SET &ERRINS# = SYS_LAST_ERR_INS()
:   SET &MESSAGE# = GET_MSG_TXT(&ERRNR#,&ERRINS#)
:   PRINT &MESSAGE#
:   SET &RET# = SEND_MAIL("meier@uc4.com",,"Fehler in der
Verarbeitung",&MESSAGE#)
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
GET_MSG_TXT	Ermittelt den Meldungstext des zuletzt aufgetretenen Fehlers.
SYS_LAST_ERR_INS	Liefert den variablen Teil der Fehlermeldung bei einem zuletzt aufgetretenen Fehler.
SYS_LAST_ERR_NR	Gibt die Fehlernummer eines zuletzt aufgetretenen Fehlers zurück.

Script-Sprachmittel - Fehlerbehandlung und Meldungen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.6.9 SEND_MAIL

Script-Funktion: Sendet E-Mail an einen Benutzer.

Syntax

SEND_MAIL*(**Empfänger**, [Cc], **Betreff**, **Text**[, Anlage])*

Syntaxteil	Beschreibung/Format

Empfänger	Nachrichtenempfänger. Format: Script-Literal oder Script-Variable Soll die E-Mail an mehrere Empfänger gesendet werden, so sind die Adressen mit Semikolon zu trennen.
Cc	Empfänger, die eine Kopie dieser Nachricht erhalten sollen. Format: Script-Literal oder Script-Variable Standard: ""
Betreff	Kurzbezeichnung der Nachricht. Format: Script-Literal oder Script-Variable
Text	Nachrichtentext. Format: Script-Literal oder Script-Variable
Anlage	Pfad und Namen der Dateien, die als Anlage geschickt werden soll. Format: Script-Literal oder Script-Variable Standard: "" Trennen Sie die einzelnen Pfade mit einem Semikolon (;), wenn Sie mehrere Dateien senden möchten.

Rückgabewerte
"0" - Die E-Mail wurde erfolgreich versendet. "10034" - Es ist kein Host mit E-Mail-Kopplung aktiv. "50006" - Der SMTP Server antwortet mit einem Fehlercode. "50012" - Zeitablauf - SMTP-Server '&02' hat nicht in '&01' Sekunden geantwortet. "50014" - Die Anlage existiert nicht. "50027" - Authentifizierung am STMP-Server fehlgeschlagen. "50028" - Die Empfänger-Adresse ist nicht gültig und wurde daher vom SMTP-Server abgelehnt. "50029" - Verbindung mit dem SMTP-Server ist nicht möglich. "50030" - Fehler bei der Socket-Erstellung. "50031" - Die Hostinformationen des SMTP-Servers konnten nicht ermittelt werden. "50032" - Kommunikation mit dem SMTP-Server ist nicht mehr möglich. "50033" - Es können keine Daten vom SMTP-Server empfangen werden. "50034" - Es können keine Daten zum SMTP-Server gesendet werden. "50035" - Initialisierung der Windows-Sockets nicht möglich. "50036" - Hostname des lokalen Rechners kann nicht ermittelt werden.

Anmerkungen

Das Senden einer E-Mail hängt von den Möglichkeiten ab, welche die PC-Konfiguration eines Benutzers bietet.

Die Script-Funktion prüft nicht, ob der angegebene *Empfänger* tatsächlich existiert. Die Nachricht wird auch an einen falschen Empfänger geschickt.

Kann die E-Mail nicht gesendet werden, da beispielsweise die Anlage nicht gefunden werden kann oder Mail-Kopplung nicht aktiv ist, wird das Script standardmäßig weiter prozessiert. Die Script-Funktion liefert in diesem Fall einen entsprechenden Rückgabewert.
Mit der Script-Anweisung :ON_ERROR haben Sie die Möglichkeit die Prozessierung des Scripts im Fehlerfall abzubrechen. Den Fehler können Sie mit den Script-Funktionen zur Fehlerbehandlung analysieren.

⚠ Beachten Sie, dass E-Mails mit diesem Sprachmittel immer über die Automation Engine gesendet werden. Dateien, die mitgeschickt werden soll, müssen sich daher auf dem Rechner der Automation Engine befinden oder von dort erreichbar sein (UNC-Pfad).

Beachten Sie, dass die Anzahl der Zeichen in einer Scriptzeile auf maximal 1024 Zeichen beschränkt ist!

Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

⚠ Das Versenden von E-Mails setzt die Konfiguration der E-Mail Kopplung voraus.

Beispiele

Im ersten Beispiel wird ein E-Mail an nur einen Empfänger (keine Kopien) verschickt. Dieses E-Mail hat eine Datei als Anlage.

```
:SET &OUT# = SEND_MAIL("meier@automic.at",,"Besprechung","Termin heute
um 14 Uhr","/automic/agenda")
```

Hier wird ein E-Mail, das zwei Anlagen enthält, an mehrere Personen gesendet.

```
:SET&OUT# = SEND_MAIL
("meier@automic.at;smith@automic.us",,"Besprechung","Termin
abgesagt","c:\AUTOMIC\agenda.doc;c:\AUTOMIC\termine.txt")
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:SEND_MSG	Schickt Nachrichten an Benutzer, die am UserInterface angemeldet sind.
:ON_ERROR	Legt die Reaktion auf bestimmte Fehler bzw. Meldungen von Script-Sprachmitteln fest.

Script-Sprachmittel - Fehlerbehandlung und Meldungen
Beispielsammlung
Ermittlung der Fehlermeldung und -nummer
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.6.10 SYS_LAST_ERR_INS

Script-Funktion: Liefert den variablen Teil der Fehlermeldung bei einem zuletzt aufgetretenen Fehler.

Syntax

SYS_LAST_ERR_INS*()*

Rückgabewerte

Variabler Teil der Meldung des zuletzt aufgetretenen Fehlers.
" " - Es ist kein Fehler aufgetreten.

Anmerkungen

Bei einigen ausgewählten Script-Sprachmitteln kann durch der Verwendung von :ON_ERROR das Script bei Fehlern trotzdem fortgesetzt werden. In diesen Fällen ist der variable Teil der Fehlermeldung mit der Funktion ermittelbar.

Die meisten Fehlermeldungen bestehen nicht nur aus einem vorgegebenen Text. Teile der Meldung beziehen sich z.B. auf Objektnamen oder Zeilen im Script. Diese variablen Teile der Fehlermeldung sind in einfache Hochkommata eingeschlossen.

Beispiel für eine Fehlermeldung:

U0020645 - Laufzeitfehler in Objekt '&01' Zeile '&02': Es konnte kein Objekt mit dem Namen '&04' gefunden werden.

Wenn die Fehlermeldung ausgegeben wird, werden die variablen Teile entsprechend ausgefüllt:

U0020645 - Laufzeitfehler in Objekt 'MOVE_OBJECT' Zeile '0003': Es konnte kein Objekt mit dem Namen 'MAWI.ABSCHLUSS.2005' gefunden werden.

Beispiel

Das Beispiel versucht die Speicherkapazität eines Laufwerkes zu prüfen. Ist dies nicht vorhanden oder kann darauf nicht zugegriffen werden, kommt es zu einem Fehler. Fehlernummer und variabler Teil der Fehlermeldung werden gelesen. Mit diesen Informationen wird die komplette Fehlermeldung ermittelt und als Nachricht an einen Benutzer verschickt.

```
:SET &CHECK#  = GET_FILESYSTEM("WIN21G", "Z:\", FILESYSTEM_SPACE_
TOTAL)
:SET &ERRINS# = SYS_LAST_ERR_INS()
:IF &ERRINS# NE ""
:   SET &ERRNR#   = SYS_LAST_ERR_NR()
:   SET &MESSAGE# = GET_MSG_TXT (&ERRNR#,&ERRINS#)
:   SEND_MSG BU,BU, &MESSAGE#
:ENDIF
```

Siehe auch:

GET_MSG_TXT	Ermittelt den Meldungstext des zuletzt aufgetretenen Fehlers.
GET_MSG_TYPE	Ermittelt den Typ einer Meldungsnummer.
SYS_LAST_ERR_NR	Gibt die Fehlernummer eines zuletzt aufgetretenen Fehlers zurück.
SYS_LAST_ERR_SYSTXT	Ermittelt den Fehlertext des Betriebssystems eines zuletzt aufgetretenen Fehlers.

Script-Sprachmittel - Fehlerbehandlung und Meldungen
Beispielsammlung
Ermittlung der Fehlermeldung und -nummer
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.6.11 SYS_LAST_ERR_NR

Script-Funktion: Gibt die Fehlernummer eines zuletzt aufgetretenen Fehlers zurück.

Syntax

SYS_LAST_ERR_NR*()*

Rückgabewerte
Fehlernummer des zuletzt aufgetretenen Fehlers. "0" - Es ist kein Fehler aufgetreten.

Anmerkungen

Bei einigen ausgewählten Script-Sprachmitteln kann durch der Verwendung von :ON_ERROR das Script bei Fehlern trotzdem fortgesetzt werden. In diesen Fällen ist die Fehlernummer mit der Funktion ermittelbar.

Beispiel

Das Beispiel versucht die Speicherkapazität eines Laufwerkes zu prüfen. Ist dies nicht vorhanden oder kann darauf nicht zugegriffen werden, kommt es zu einem Fehler. Fehlernummer und variabler Teil der Fehlermeldung werden gelesen. Mit diesen Informationen wird die komplette Fehlermeldung ermittelt und als Nachricht an einen Benutzer verschickt.

```
:SET &CHECK# = GET_FILESYSTEM("WIN21G", "Z:\", FILESYSTEM_SPACE_TOTAL)
:SET &ERRNR# = SYS_LAST_ERR_NR()
:IF &ERRNR# NE "0"
:   SET &ERRINS#  = SYS_LAST_ERR_INS()
:   SET &MESSAGE# = GET_MSG_TXT (&ERRNR#,&ERRINS#)
:   SEND_MSG BU,BU, &MESSAGE#
:ENDIF
```

Siehe auch:

GET_MSG_TXT	Ermittelt den Meldungstext des zuletzt aufgetretenen Fehlers.
GET_MSG_TYPE	Ermittelt den Typ einer Meldungsnummer.
SYS_LAST_ERR_INS	Liefert den variablen Teil der Fehlermeldung bei einem zuletzt aufgetretenen Fehler.
SYS_LAST_ERR_ SYSTXT	Ermittelt den Fehlertext des Betriebssystems eines zuletzt aufgetretenen Fehlers.

Script-Sprachmittel - Fehlerbehandlung und Meldungen
Beispielsammlung
Ermittlung der Fehlermeldung und -nummer
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.6.12 SYS_LAST_ERR_SYSTXT

Script-Funktion: Ermittelt den Fehlertext des Betriebssystems eines zuletzt aufgetretenen Fehlers.

Syntax

SYS_LAST_ERR_SYSTXT*()*

Rückgabewerte
Fehlertext des Betriebssystem eines zuletzt aufgetretenen Fehlers. " " - Es ist kein Fehler aufgetreten.

Anmerkungen

Mit der Script-Funktion wird der im Fehlerfall vom Betriebssystem ausgegebene Fehlertext ermittelt. Der Rückgabewert der Script-Funktion ist dieser Fehlertext oder ein Leerstring, wenn kein Fehler auftrat.

Beispiel

Das Beispiel versucht die Speicherkapazität eines Laufwerkes zu prüfen. Ist dies nicht vorhanden, kommt es zu einem Fehler. Der Fehlertext des Betriebssystems ("Das System kann den angegebenen Pfad nicht finden.") wird dann als Nachricht an einen Benutzer verschickt.

```
:SET  &CHECK# = GET_FILESYSTEM("WIN21G", "Z:\", FILESYSTEM_SPACE_
TOTAL)
 :SET  &MESSAGE# = SYS_LAST_ERR_SYSTXT()
 :IF  &MESSAGE# NE ""
 :   SEND_MSG BU,BU,"Systemfehler: &MESSAGE#"
 :ENDIF
```

Siehe auch:

SYS_LAST_ERR_NR	Gibt die Fehlernummer eines zuletzt aufgetretenen Fehlers zurück.

Script-Sprachmittel - Fehlerbehandlung und Meldungen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7 Aktivierungsdaten

3.7.1 GET_PARENT_NAME

Script-Funktion: Gibt den Namen der übergeordneten Aufgabe (Parent) zurück.

Syntax

GET_PARENT_NAME*(RunID *[, Aktivierungstyp]**)***

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) der untergeordneten Aufgabe.
Aktivierungstyp	Erlaubte Werte: ACT = Activator PRC = Processor (Standardwert)

Rückgabewerte
Name der übergeordneten Aufgabe. "*SCRIPT" - Die Aufgabe wurde über ein CallAPI aktiviert. " " - Es existiert keine übergeordnete Aufgabe (nur bei Aktivierungstyp PRC).

Anmerkungen

Mit der Script-Funktion ermitteln Sie den Namen der übergeordneten Aufgabe eines Objektes, das der Objektklasse der aktivierbaren Objekte angehört. Bei einer manuellen Aktivierung ist dies der Name des Benutzer-Objektes (z.B. MEIER/DEV).

Geben Sie keinen Aktivierungstyp an, ermittelt die Script-Funktion den Processor.

⚠ Damit der Name einer Gruppe ermittelt werden kann, muss bei der Aufgabe, welche die Script-Funktion verwendet, "Zur Laufzeit generieren" (Registerkarte **Attribute**) aktiviert sein. Sonst kann der Name nur bei Jobs im Post-Script festgestellt werden.

Beispiel

Im folgenden Beispiel wird die laufende Nummer einer Aufgabe ermittelt und anschließend ihr Activator bestimmt.

```
:SET  &RunID = GET_UC_OBJECT_NR(MAWI.ABSCHLUSS)
:SET  &RET# = GET_PARENT_NAME(&RunID, ACT)
:PRINT  "Der Name der übergeordneten Aufgabe lautet: &RET#"
```

Siehe auch:

Script-Sprachmittel	Beschreibung

GET_PARENT_NR	Gibt die laufende Nummer (RunID) der übergeordneten Aufgabe (Parent) zurück.
GET_PARENT_ TYPE	Gibt den Objekttyp der übergeordneten Aufgabe (Parent) zurück.

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7.2 GET_PARENT_NR

Script-Funktion: Gibt die laufende Nummer (RunID) der übergeordneten Aufgabe (Parent) zurück.

Syntax

GET_PARENT_NR*(RunID *[, Aktivierungstyp]**)***

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) der untergeordneten Aufgabe.
Aktivierungstyp	Erlaubte Werte: ACT = Activator PRC = Processor (Standard)

Rückgabewerte
RunID der übergeordneten Aufgabe. Session-ID bei einer manuellen Aktivierung (nur bei Aktivierungstyp ACT) " " - Es existiert keine übergeordnete Aufgabe (nur bei Aktivierungstyp PRC).

Anmerkungen

Mit der Script-Funktion ermitteln Sie die laufende Nummer (RunID) der übergeordneten Aufgabe eines Objektes, das der Objektklasse der aktivierbaren Objekte angehört. Bei einer manuellen Aktivierung ist dies die Session-ID des Benutzers.

Geben Sie keinen Aktivierungstyp an, ermittelt die Script-Funktion den Processor.

⚠ Damit die RunID einer Gruppe ermittelt werden kann, muss bei der Aufgabe, welche die Script-Funktion verwendet, "Zur Laufzeit generieren" (Registerkarte **Attribute**) aktiviert sein. Sonst kann die RunID nur bei Jobs im Post-Script festgestellt werden.

Beispiel

Im folgenden Beispiel wird die laufende Nummer einer Aufgabe ermittelt und anschließend jene ihres Activators bestimmt.

```
:SET  &RunID = GET_UC_OBJECT_NR(MAWI.ABSCHLUSS)
:SET  &RET# = GET_PARENT_NR(&RunID, ACT)
:PRINT  "Die RunID der übergeordneten Aufgabe lautet: &RET#"
```

Siehe auch:

Script-Sprachmittel	Beschreibung
GET_PARENT_NAME	Gibt den Namen der übergeordneten Aufgabe (Parent) zurück.
GET_PARENT_TYPE	Gibt den Objekttyp der übergeordneten Aufgabe (Parent) zurück.

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7.3 GET_PARENT_TYPE

Script-Funktion: Gibt den Objekttyp der übergeordneten Aufgabe (Parent) zurück.

Syntax

GET_PARENT_TYPE(***RunID*** *[, Aktivierungstyp]*)

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) der untergeordneten Aufgabe.
Aktivierungstyp	Erlaubte Werte: ACT = Activator PRC = Processor (Standardwert)

Rückgabewerte
Objekttyp der übergeordneten Aufgabe. "USER" - Bei einer manuellen Aktivierung (nur bei Aktivierungstyp ACT). "API" - Bei einer Aktivierung über CallAPI (nur bei Aktivierungstyp ACT). " " - Es existiert keine übergeordnete Aufgabe (nur bei Aktivierungstyp PRC).

Anmerkungen

Mit der Script-Funktion ermitteln Sie den Objekttyp der übergeordneten Aufgabe eines Objektes, das der Objektklasse der aktivierbaren Objekte angehört. Bei einer manuellen Aktivierung ist dies "USER".

Geben Sie keinen Aktivierungstyp an, ermittelt die Script-Funktion den Processor.

⚠ Damit der Objekttyp einer Gruppe ermittelt werden kann, muss bei der Aufgabe, welche die Script-Funktion verwendet, "Zur Laufzeit generieren" (Registerkarte **Attribute**) aktiviert sein. Sonst kann der Objekttyp nur bei Jobs im Post-Script festgestellt werden.

Beispiel

Im folgenden Beispiel wird die laufende Nummer einer Aufgabe ermittelt und anschließend der Objekttyp ihres Activator bestimmt.

```
:SET  &RunID# = GET_UC_OBJECT_NR(MAWI.ABSCHLUSS)
:SET  &RET# = GET_PARENT_TYPE(&RunID#, ACT)
:PRINT  "Typ der übergeordneten Aufgabe: &RET#"
```

Siehe auch:

Script-Sprachmittel	Beschreibung
GET_PARENT_NAME	Gibt den Namen der übergeordneten Aufgabe (Parent) zurück.
GET_PARENT_NR	Gibt die laufende Nummer (RunID) der übergeordneten Aufgabe (Parent) zurück.

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7.4 GET_UC_OBJECT_STATUS

Script-Funktion: Liefert den Status eines aktivierten Objektes.

Syntax

GET_UC_OBJECT_STATUS(*[[Objekttyp], [RunID], [Anforderung]]*)

Syntaxteil	Beschreibung/Format
Objekttyp	Kurzbezeichnung des aktivierbaren Objekttyps Format: AE-name, Script-Literal oder Script-Variable Optionaler Parameter, da der Objekttyp durch die laufende Nummer (RunID) eindeutig zugeordnet werden kann (kompatibel zur Version 2.6xx).
RunID	Laufende Nummer (RunID) des aktivierten Objektes Format: Script-Literal oder Script-Variable

Anforderung	Information, die zum aktivierten Objekt ermittelt werden soll Format: Script-Literal oder Script-Variable Erlaubte Werte: "STATUS" (Standardwert), "RETCODE", "STATUS_TEXT" "RETCODE" = Fordert den Rückgabewert des aktivierten Objektes an. "STATUS" = Fordert den aktuellen System-Rückgabewert des aktivierten Objektes an. "STATUS_TEXT" = Fordert den Statustext an. Dieser Wert ist nur in Jobs auf Betriebssystem-Agenten verfügbar. In SAP-Jobs kann der Statustext beispielsweise nicht ermittelt werden.

Rückgabewerte
Je nach Angabe entweder der Rückgabewert, der Status oder der Statustext der Aufgabe. " " - Die angegebene Aufgabe existiert nicht.

Anmerkungen

Mit der Script-Funktion stellen Sie den Status eines Objektes fest, das der Objektklasse der aktivierbaren Objekte angehört.

⚠ Die Script-Funktion liefert auch einen Wert, wenn die Aufgabe nicht mehr aktiv ist. Er wird aus der Statistik entnommen. Falls keine Statistik gefunden wird, ist der Rückgabewert ein Leerstring.

Alle Parameter der Script-Funktion sind optional. Wenn Sie nur den Parameter *Anforderung* verwenden, sind trotzdem die beiden Kommata zu setzen. Beispiel:

```
:SET &STATUS# = GET_UC_OBJECT_STATUS( ,,"STATUS")
```

Wird *Objekttyp* **und** *laufende Nummer (RunID)* nicht angegeben, liefert die Script-Funktion den Status der eigenen Aufgabe. Es wird also der Status der Aufgabe, welche die Script-Funktion verwendet, zurückgegeben.

Wenn *Objekttyp* **oder** *laufende Nummer (RunID)* nicht angegeben werden, gelten die folgenden Besonderheiten.

- Wird nur *Objekttyp* als Parameter angegeben, ist der Rückgabewert der Status der eigenen Aufgabe. Dabei muss *Objekttyp* dem Objekttyp der eigenen Aufgabe entsprechen.
- Wird nur *laufende Nummer (RunID)* angegeben, kann damit entweder der Status der eigenen Aufgabe oder der Status einer anderen Aufgabe ermittelt werden.
- Wird die laufende Nummer einer anderen Aufgabe an die Script-Funktion übergeben, müssen beide Aufgaben vom gleichen Objekttyp sein. Ist das nicht der Fall, muss der Objekttyp der anderen Aufgabe explizit angegeben werden.

Es wird ein Leerstring zurückgegeben, wenn *Objekttyp* und *laufende Nummer (RunID)* nicht zusammenpassen. Es kommt auch ein Leerstring zurück, wenn keine Aufgabe zur angegebenen laufende Nummer (RunID) gefunden wurde.

Wird *Anforderung* nicht verwendet, liefert die Script-Funktion den Statuscode zurück.

Für Jobs kann auch der Statustext ermittelt werden. Das ist der Text, den der Jobmelder im Trailer ausgegeben hat oder welcher mit :MODIFY_STATE geändert wurde. Bei den anderen Objekttypen wird ein Leerstring zurückgegeben.

Beispiel

Im ersten Beispiel wird der Job "DB.USE" aktiviert, um danach seinen Status zu ermitteln. Um die laufende Nummer des Jobs zu erhalten, wird eine weitere Script-Funktion eingesetzt.

```
:SET &JNR# = ACTIVATE_UC_OBJECT(JOBS,DB.USE)
:SET &STATUS# = GET_UC_OBJECT_STATUS(JOBS,&JNR#)
:PRINT "Der Status von Job(&JNR#) ist &STATUS#"
```

Das zweite Beispiel ermittelt den eigenen Status einer Aufgabe. Der Status wird im Aktivierungsprotokoll ausgegeben.

```
:SET &RET# = GET_UC_OBJECT_STATUS()
:PRINT &RET#
```

Im dritten Beispiel soll im Script eines Jobs der Status des Workflows (Parent) ermittelt werden, in welchem dieser Job läuft. Da unterschiedliche Objekttypen vorliegen, müssen der Script-Funktion Objekttyp und laufende Nummer (RunID) des Workflows übergeben werden.

```
:SET &RUNNR# = SYS_ACT_PARENT_NR()
:SET &STATUS# = GET_UC_OBJECT_STATUS (JOBP,&RUNNR#)
:SEND_MSG BU,BU,"Der Status des Workflows ist: &STATUS#."
```

Das vierte Beispiel liefert den Statustext eines Jobs.

```
:SET &RET# = GET_UC_OBJECT_STATUS(,,"STATUS_TEXT")
:PRINT &RET#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:MODIFY_STATE	Ändert den Rückgabewert oder Statustext eines Jobs nach dessen Ende.

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7.5 GET_UC_OBJECT_NR

Script-Funktion: Liefert die laufende Nummer (RunID) eines aktivierten Objektes.

Syntax

GET_UC_OBJECT_NR*(Objektname)*

Syntaxteil	Beschreibung/Format
Objektname	Name eines aktivierbaren Objektes. Format: Script-Literal oder Script-Variable

Rückgabewert
Laufende Nummer des aktivierten Objektes. " " - Das angegebene Objekt ist nicht aktiv.

Anmerkungen

Mit der Script-Funktion GET_UC_OBJECT_NR können Sie die laufende Nummer (RunID) eines Objektes ermitteln, welches der Objektklasse der aktivierbaren Objekte angehört.

Wurde das Objekt mehrfach aktiviert, wird die älteste laufende Nummer (RunID) zurückgegeben.

Beispiel

Im folgenden Beispiel wird geprüft, ob die Aufgabe aktiv ist.

```
:SET &RunID# = GET_UC_OBJECT_NR(&aufgabe#)

 :IF &RunID# = ""
:   STOP MSG,50,"&aufgabe# ist nicht aktiv!"
 :ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
GET_PARENT_NR	Gibt die laufende Nummer (RunID) der übergeordneten Aufgabe (Parent) zurück.
SYS_ACT_TOP_NR	Ermittelt die laufende Nummer (RunID) des obersten Workflows.

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7.6 SYS_ACT_HOST

Script-Funktion: Ermittelt den Namen des Hosts.

Syntax

SYS_ACT_HOST*()*

Rückgabewert
Name des Agenten des Objektes in dem die Script-Funktion aufgerufen wird.

Anmerkungen

Die Script-Funktion kann in Ereignissen des Typs "Dateisystem" und "Konsole" (Registerkarte Script und !Script) sowie in den Script-Registerkarten von Jobs verwendet werden.

Beachten Sie, dass der Name des Host zurückgegeben wird, für den das Ereignis oder der Job letztlich ausgeführt wird. Dieser kann vom definierten Host abweichen, wenn beispielsweise der

Host eines Jobs im Attribut-Dialog vom Benutzer geändert oder durch ein :PUT_ATT im Pre-Script gesetzt wird.

Beispiel

Das Beispiel zeigt, wie der ermittelte Hostname einer Script-Variablen zugewiesen wird. Diese Script-Variable wird anschließend als erster Parameter der Script-Funktion GET_FILESYSTEM verwendet, um Dateien in einem temporären Verzeichnis zu zählen.

```
:SET &HOST# = SYS_ACT_HOST()
 :SET &ANZAHL# = GET_FILESYSTEM(&HOST#,"c:\temp\*.exe","PATH_FILE_
COUNT")
```

Siehe auch:

SYS_HOST_ALIVE	Überprüft, ob ein bestimmter Host aktiv ist.

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7.7 SYS_ACT_JP

Script-Funktion: Ermittelt, ob die Aufgabe aus einem Workflow heraus aktiviert wurde.

Syntax

SYS_ACT_JP*()*

Rückgabewerte
"Y" - Aufgabe wurde von einem Workflow aktiviert. "N" - Die Aufgabe wurde nicht von einem Workflow aktiviert.

Beispiel

Im Beispiel wurde eine Aufgabe manuell gestartet. Der Rückgabewert ist folglich "N" und führt dazu, dass "Aufgabe läuft nicht im Workflow" im Aktivierungsprotokoll ausgegeben wird.

```
:SET &RET# = SYS_ACT_JP()
 :IF &RET# = "Y"
:   PRINT "Aufgabe läuft im Workflow"
 :ELSE
:   PRINT "Aufgabe läuft nicht im Workflow"
 :ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung

SYS_ACT_PARENT_NAME	Gibt den Namen der übergeordneten Aufgabe zurück.
SYS_ACT_PARENT_NR	Gibt die laufende Nummer (RunID) der übergeordneten Aufgabe zurück.

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7.8 SYS_ACT_ME_NAME

Script-Funktion: Liefert den Namen des eigenen Objektes.

Syntax

SYS_ACT_ME_NAME *()*

Rückgabewert
Name des Objektes in dem die Script-Funktion aufgerufen wird.

Beispiel

Im Post-Script eines Jobs wird dessen Name und Status ermittelt. Endet der Job nicht normal (ENDED_OK - 1900), wird der Jobabbruch per E-Mail gemeldet.

```
:SET &NAME#   = SYS_ACT_ME_NAME()
:SET &STATUS# = GET_UC_OBJECT_STATUS()
:IF &STATUS#  < "1900"
:   SET &OUT# = SEND_MAIL("meier@automic.at",,"Jobabbruch!","Job:
&NAME# ist abgebrochen!")
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_ACT_ME_NR	Liefert die laufende Nummer (RunID) des eigenen Objektes.
SYS_ACT_ME_TYPE	Liefert den Objekttyp des eigenen Objektes.

Script-Sprachmittel - Aktivierungsdaten
Beispielsammlung
Ermittlung der Fehlermeldung und -nummer
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7.9 SYS_ACT_ME_NR

Script-Funktion: Liefert die laufende Nummer (RunID) des eigenen Objektes.

Syntax

SYS_ACT_ME_NR*()*

Rückgabewert
Laufende Nummer des Objektes in dem die Script-Funktion aufgerufen wird.

Anmerkungen

Bei einem Wiederanlauf der Aufgabe mit Referenz-RunID, liefert die Script-Funktion diese Referenz-RunID und nicht die RunID der Durchführung. Verwenden Sie SYS_ACT_RESTART_ME_NR, um stets die laufende Nummer des im Wiederanlauf-Modus aktivierten Objektes zu erhalten.

Beispiel

Im Beispiel wird die RunID einer Aufgabe dazu verwendet, um Informationen zum Parent zu erlangen. Wurde die Aufgabe durch einen Benutzer gestartet, wird dessen Name im Aktivierungsprotokoll ausgegeben. Erfolgte der Start durch eine Aufgabe, ist es der Name dieser Aufgabe.

```
:SET &RUNID# = SYS_ACT_ME_NR()
 :SET &PNAME# = GET_PARENT_NAME(, &RUNID#, "ACT")
 :PRINT "Parent-Name: &PNAME#"
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_ACT_ME_NAME	Liefert den Namen des eigenen Objektes.
SYS_ACT_ME_TYPE	Liefert den Objekttyp des eigenen Objektes.

Script-Sprachmittel - Aktivierungsdaten
Beispielsammlung
Ermittlung der Fehlermeldung und -nummer
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7.10 SYS_ACT_ME_TYPE

Script-Funktion: Liefert den Objekttyp des eigenen Objektes.

Syntax

SYS_ACT_ME_TYPE*()*

Rückgabewert
Kurzzeichen des Objekttyps des Objektes in dem die Script-Funktion aufgerufen wird.

Anmerkungen

Die Script-Funktion ermittelt den Objekttyp (Kurzform) des eigenen Objektes.

Beispiel

Im Beispiel wird SYS_ACT_ME_TYPE im Script eines Jobs verwendet und der Rückgabewert "JOBS" im Aktivierungsprotokoll ausgegeben.

```
:SET &RET# = SYS_ACT_ME_TYPE()
 :PRINT "Objekttyp ist: &RET#"
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_ACT_ME_NAME	Liefert den Namen des eigenen Objektes.
SYS_ACT_ME_NR	Liefert die laufende Nummer (RunID) des eigenen Objektes.

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7.11 SYS_ACT_PARENT_NAME

Script-Funktion: Liefert den Namen der übergeordneten Aufgabe.

Syntax

SYS_ACT_PARENT_NAME *([Aktivierungstyp])*

Syntaxteil	Beschreibung/Format

Aktivierungstyp	Erlaubte Werte: ACT = Activator PRC = Processor (Standardwert)

Rückgabewerte
Name der übergeordneten Aufgabe. "*SCRIPT" - Die Aufgabe wurde über ein CallAPI aktiviert. " " - Es existiert keine übergeordnete Aufgabe (nur bei Aktivierungstyp PRC).

Anmerkungen

Die Script-Funktion ermittelt den Namen der übergeordneten Aufgabe (Parent). Bei einer manuellen Aktivierung ist dies der Name des Benutzer-Objektes (z.B. MEIER/DEV).

Geben Sie keinen Aktivierungstyp an, ermittelt die Script-Funktion den Processor.

⚠ Damit der Name einer Gruppe ermittelt werden kann, muss bei der Aufgabe, welche die Script-Funktion verwendet, "Zur Laufzeit generieren" (Registerkarte **Attribute**) aktiviert sein. Sonst kann der Name nur bei Jobs im Post-Script festgestellt werden.

Beispiele

Im ersten Beispiel wird geprüft, ob die Aufgabe in einer Gruppe, einem Workflow oder einem Schedule läuft. Ist dies der Fall, wird der Name der Aufgabe in das Aktivierungsprotokoll geschrieben. Gibt es keine übergeordnete Aufgabe, wird darauf mit einem Eintrag im Aktivierungsprotokoll hingewiesen.

```
:SET &NAME# = SYS_ACT_PARENT_NAME()
:IF &NAME# NE " "
:   PRINT "Name von Parent ist &NAME#."
:ELSE
:   PRINT "Kein Parent vorhanden"
:ENDIF
```

Das zweite Beispiel ermittelt den Activator der Aufgabe. Im Aktivierungsprotokoll wird dessen Name ausgegeben. Wurde die Aufgabe manuell aktiviert, enthält das Aktivierungsprotokoll Namen und Abteilung des jeweiligen Benutzers.

```
:SET &NAME# = SYS_ACT_PARENT_NAME(ACT)
:PRINT "Aufgabe wurde aktiviert durch &NAME#."
```

Im dritten Beispiel wird auf dieselbe Weise wie im zweiten der Activator einer Aufgabe ermittelt. Bei einer Aktivierung über ein CallAPI wird beispielsweise eine solche Zeile im Aktivierungsprotokoll ausgegeben:

```
2004-01-27 13:19:36 - U0020408 Aufgabe wurde aktiviert durch *SCRIPT.
```

Siehe auch:

Script-Sprachmittel	Beschreibung

SYS_ACT_PARENT_NR	Liefert die laufende Nummer (RunID) der übergeordneten Aufgabe.
SYS_ACT_PARENT_ TYPE	Liefert den Objekttyp der übergeordneten Aufgabe.

Script-Sprachmittel - Aktivierungsdaten
Beispielsammlung
Benachrichtigung mit variablem Meldungstext
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7.12 SYS_ACT_PARENT_NR

Script-Funktion: Liefert die laufende Nummer (RunID) der übergeordneten Aufgabe.

Syntax

SYS_ACT_PARENT_NR *([Aktivierungstyp])*

Syntaxteil	Beschreibung/Format
Aktivierungstyp	Erlaubte Werte: ACT = Activator PRC = Processor (Standardwert)

Rückgabewerte
RunID der übergeordneten Aufgabe. Session-ID bei einer manuellen Aktivierung (nur bei Aktivierungstyp ACT). " " - Es existiert keine übergeordnete Aufgabe (nur bei Aktivierungstyp PRC).

Anmerkungen

Die Script-Funktion ermittelt die laufende Nummer (RunID) der übergeordneten Aufgabe (Parent). Bei einer manuellen Aktivierung ist dies die Session-ID des Benutzers.

Geben Sie keinen Aktivierungstyp an, ermittelt die Script-Funktion den Processor.

⚠ Damit die RunID einer Gruppe ermittelt werden kann, muss bei der Aufgabe, welche die Script-Funktion verwendet, "Zur Laufzeit generieren" (Registerkarte **Attribute**) aktiviert sein. Sonst kann die RunID nur bei Jobs im Post-Script festgestellt werden.

Beispiele

Im ersten Beispiel wird geprüft, ob die Aufgabe in einer Gruppe, einem Workflow oder einem Schedule läuft. Ist dies der Fall, wird die laufende Nummer (RunID) der Aufgabe in das Aktivierungsprotokoll geschrieben. Gibt es keine übergeordnete Aufgabe, wird darauf mit einem Eintrag im Aktivierungsprotokoll hingewiesen.

```
:SET  &NR# = SYS_ACT_PARENT_NR()
:IF  &NR#  = " "
:   PRINT  "Kein Parent vorhanden"
:ELSE
:   PRINT  "RunID von Parent ist &NR#."
:ENDIF
```

Das zweite Beispiel ermittelt den Activator der Aufgabe. Im Aktivierungsprotokoll wird die laufende Nummer (RunID) einer Aufgabe ausgegeben, wenn die Aktivierung durch eine Aufgabe erfolgte. Wurde die Aufgabe manuell aktiviert, enthält das Aktivierungsprotokoll die RunID der jeweiligen Benutzersession.

```
:SET  &NR# = SYS_ACT_PARENT_NR(ACT)
:PRINT  "RunID von Parent ist &NR#."
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_ACT_PARENT_NAME	Liefert den Namen der übergeordneten Aufgabe.
SYS_ACT_PARENT_TYPE	Liefert den Objekttyp der übergeordneten Aufgabe.

Script-Sprachmittel - Aktivierungsdaten
Beispielsammlung
Benachrichtigung mit variablem Meldungstext

Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7.13 SYS_ACT_PARENT_TYPE

Script-Funktion: Liefert den Objekttyp der übergeordneten Aufgabe.

Syntax

SYS_ACT_PARENT_TYPE*(Aktivierungstyp)*

Syntaxteil	Beschreibung/Format
Aktivierungstyp	Erlaubte Werte: ACT = Activator PRC = Processor (Standardwert)

Rückgabewerte
Objekttyp der übergeordneten Aufgabe. "USER" - Bei einer manuellen Aktivierung (nur bei Aktivierungstyp ACT). "API" - Bei einer Aktivierung über CallAPI (nur bei Aktivierungstyp ACT). " " - Es existiert keine übergeordnete Aufgabe (nur bei Aktivierungstyp PRC).

Anmerkungen

Die Script-Funktion ermittelt den Objekttyp der übergeordneten Aufgabe (Parent). Bei einer manuellen Aktivierung ist dies "USER".

Geben Sie keinen Aktivierungstyp an, ermittelt die Script-Funktion den Processor.

⚠ Damit der Objekttyp einer Gruppe ermittelt werden kann, muss bei der Aufgabe, welche die Script-Funktion verwendet, "Zur Laufzeit generieren" (Registerkarte **Attribute**) aktiviert sein. Sonst kann der Objekttyp nur bei Jobs im Post-Script festgestellt werden.

Beispiele

Im ersten Beispiel wird geprüft, ob die Aufgabe in einer Gruppe, einem Workflow oder einem Schedule läuft. Ist dies der Fall, wird der Objekttyp der Aufgabe in das Aktivierungsprotokoll geschrieben. Gibt es keine übergeordnete Aufgabe, wird darauf mit einem Eintrag im Aktivierungsprotokoll hingewiesen.

```
:SET  &TYPE# = SYS_ACT_PARENT_TYPE()
:IF  &TYPE#  = " "
:   PRINT  "Kein Parent vorhanden"
:ELSE
:   PRINT  "Objekttyp von Parent ist &TYPE#."
:ENDIF
```

Das zweite Beispiel ermittelt den Activator der Aufgabe. Im Aktivierungsprotokoll wird der Objekttyp einer Aufgabe ausgegeben, wenn die Aktivierung durch eine Aufgabe erfolgte. Wurde die Aufgabe manuell aktiviert, enthält das Aktivierungsprotokoll den Eintrag "Aufgabe wurde aktiviert durch USER".

```
:SET  &TYPE# = SYS_ACT_PARENT_TYPE(ACT)
:PRINT  "Aufgabe wurde aktiviert durch &TYPE#."
```

Im dritten Beispiel wird auf die selbe Weise wie im zweiten der Activator einer Aufgabe ermittelt. Bei einer Aktivierung über die CallAPI wird beispielsweise eine solche Zeile im Aktivierungsprotokoll ausgegeben:

```
2004-01-28 11:19:26 - U0020408 Aufgabe wurde aktiviert durch API.
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_ACT_PARENT_NAME	Liefert den Namen der übergeordneten Aufgabe.
SYS_ACT_PARENT_NR	Liefert die laufende Nummer (RunID) der übergeordneten Aufgabe.

Script-Sprachmittel - Aktivierungsdaten
Objekttypen
Beispielsammlung
Benachrichtigung mit variablem Meldungstext
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7.14 SYS_ACT_PREV_NAME

Script-Funktion: Liefert den Namen der vorhergehenden Aufgabe in einem Workflow.

Syntax

SYS_ACT_PREV_NAME*()*

Rückgabewerte
Name der Vorgänger-Aufgabe "START" - Die Aufgabe ist die erste im Workflow, da es sich beim Vorgänger um den Startpunkt handelt.

Anmerkungen

Nur Aufgaben, die in Workflows laufen, dürfen diese Script-Funktion enthalten.

Die Script-Funktion ist ebenfalls nur erlaubt, wenn es genau eine vorhergehende Aufgabe gibt. Existiert mehr als ein direkter Vorgänger, kommt es zu einem Laufzeitfehler und zum Abbruch des Scripts.

Beispiel

Im Aktivierungsprotokoll wird der Name der in einem Workflow vorhergehenden Aufgabe ausgegeben.

```
:SET &NAME# = SYS_ACT_PREV_NAME()
 :PRINT "Name der vorhergehenden Aufgabe ist &NAME#."
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_ACT_PREV_NR	Liefert die laufende Nummer (RunID) der vorhergehenden Aufgabe in einem Workflow.

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7.15 SYS_ACT_PREV_NR

Script-Funktion: Liefert die laufende Nummer (RunID) der vorhergehenden Aufgabe in einem Workflow.

Syntax

SYS_ACT_PREV_NR*()*

Rückgabewerte
RunID der Vorgänger-Aufgabe "0" - Die Aufgabe ist die erste im Workflow, da es sich beim Vorgänger um den Startpunkt handelt.

Anmerkungen

Nur Aufgaben, die in Workflows laufen, dürfen diese Script-Funktion enthalten.

Die Script-Funktion ist ebenfalls nur erlaubt, wenn es genau eine vorhergehende Aufgabe gibt. Existiert mehr als ein direkter Vorgänger, kommt es zu einem Laufzeitfehler und zum Abbruch des Scripts.

Beispiel

Im Aktivierungsprotokoll wird die laufende Nummer (RunID) der in einem Workflow vorhergehenden Aufgabe ausgegeben.

```
:SET &NR# = SYS_ACT_PREV_NR()
 :PRINT "RunID der vorhergehenden Aufgabe ist &NR#."
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_ACT_PREV_NAME	Liefert den Namen der vorhergehenden Aufgabe in einem Workflow.

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7.16 SYS_ACT_PTTYP

Script-Funktion: Liefert Partnertyp des Benutzers.

Syntax

SYS_ACT_PTTYP*()*

Rückgabewerte

"D" - Aktivierung durch das UserInterface
"C" - Aktivierung durch ein CallAPI
"A" - Aktivierung durch das Java Application Interface oder Deployment Descriptor

Anmerkungen

Die Script-Funktion ermittelt, auf welche Art eine Aufgabe aktiviert wurde.

Wird die Funktion im Post-Script verwendet oder ist "Zur Laufzeit generieren" gesetzt, wird ein leerer Rückgabewert geliefert.

Beispiel

Das Beispiel zeigt, wie bei Aktivierung über die CallAPI eine Hinweiszeile in das Aktivierungsprotokoll geschrieben wird.

```
:SET &PTTYP# = SYS_ACT_PTTYP()
:IF &PTTYP#  = "C"
:   PRINT "Aufgabe wurde über CallAPI aktiviert."
:ENDIF
```

Siehe auch:

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7.17 SYS_ACT_RESTART

Script-Funktion: Ermittelt, ob Objekt im Wiederanlauf-Modus aktiviert wurde.

Syntax

SYS_ACT_RESTART*()*

Rückgabewerte

"Y" - Das Objekt wurde im Wiederanlauf-Modus aktiviert.
"N" - Es handelt sich um keinen Wiederanlauf.

Beispiel

Im Beispiel werden Verarbeitungsschritte, welche in einem Include hinterlegt sind, nur dann ausgeführt, wenn die Aufgabe im Wiederanlauf durchgeführt wird.

```
:IF SYS_ACT_RESTART() = "Y"
:   INCLUDE "INCL.RESTART.PROC"
 :ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:RESTART	Setzt Wiederanlauf-Punkte in einem aktivierbaren Objekt.
SYS_ACT_RESTART_ME_NR	Liefert die laufende Nummer (RunID) des im Wiederanlauf-Modus aktivierten Objektes.

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7.18 SYS_ACT_RESTART_COUNT

Script-Funktion: Liefert die Anzahl der Wiederanläufe von Workflow-Aufgaben, die über die Anweisung RESTART TASK durchgeführt wurden (Post-Conditions)

Syntax

SYS_ACT_RESTART_COUNT*()*

Rückgabewert
Anzahl der Wiederanläufe

Anmerkungen

Mit dem Script-Sprachmittel wird ermittelt, wie oft eine Aufgabe innerhalb eines Workflows über die Anweisung RESTART TASK bereits neu gestartet wurde. Diese Anweisung kann über die Aufgaben-Eigenschaften in der Registerkarte "Post-Conditions" definiert werden.

Dieser Wert ist ebenfalls über die vordefinierte Variable &$RESTART_COUNT# abrufbar.

Beispiel

In diesem Beispiel wird die Anzahl der bisherigen Wiederanläufe der Workflow-Aufgabe abgefragt und die Information im Aktivierungsprotokoll ausgegeben.

```
:SET  &RCOUNT# = SYS_ACT_RESTART_COUNT()
:IF  &RCOUNT# = 0
:   PRINT  "Es wurde noch kein Wiederanlauf der Aufgabe durchgeführt."
:ELSE
:   PRINT  "Anzahl der Wiederanläufe: &RCOUNT#"
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:RESTART	Setzt Wiederanlauf-Punkte in einem aktivierbaren Objekt.
SYS_ACT_RESTART	Ermittelt, ob Objekt im Wiederanlauf-Modus aktiviert wurde.
SYS_ACT_RESTART_ME_NR	Liefert die laufende Nummer (RunID) des im Wiederanlauf-Modus aktivierten Objektes.

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7.19 SYS_ACT_RESTART_ME_NR

Script-Funktion: Liefert die laufende Nummer (RunID) des im Wiederanlauf-Modus aktivierten Objektes.

Syntax

SYS_ACT_RESTART_ME_NR*()*

Rückgabewert
Laufende Nummer des Objektes in dem die Script-Funktion aufgerufen wird.

Anmerkungen

⚠ Die Script-Funktion gibt auch die eigene RunID zurück, wenn die Aufgabe normal aktiviert wurde. Sie kann also verwendet werden, ohne dass zuvor auf normale Aktivierung oder Wiederanlauf geprüft werden muss. Beachten Sie bitte das davon abweichende Verhalten von SYS_ACT_ME_NR im Wiederanlauf-Modus.

Beispiel

Im Beispiel wird die laufende Nummer (RunID) einer Aufgabe in das Aktivierungsprotokoll geschrieben. Als zusätzliche Information wird mitgegeben, ob die Aufgabe normal oder im Wiederanlauf durchgeführt wurde.

```
:SET  &RUNID# = SYS_ACT_RESTART_ME_NR()
:IF  SYS_ACT_RESTART = "Y"
:   PRINT  "Job mit RunID &RUNID# im Wiederanlauf geführt."
:ELSE
:   PRINT  "Job mit RunID &RUNID# normal durchgeführt."
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung

:RESTART	Setzt Wiederanlauf-Punkte in einem aktivierbaren Objekt.
SYS_ACT_RESTART	Ermittelt, ob Objekt im Wiederanlauf-Modus aktiviert wurde.

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7.20 SYS_ACT_TOP_NAME

Script-Funktion: Liefert den Namen des obersten Workflows.

Syntax

SYS_ACT_TOP_NAME*()*

Rückgabewerte
Name des obersten Workflows " " - Die Aufgabe wurde nicht durch einen Workflow aktiviert.

Beispiel

Im Beispiel werden die Rückgabewerte der Script-Funktionen verglichen um festzustellen, ob die Aufgabe ohne Workflow, innerhalb eines Workflows oder im Rahmen verschachtelter Workflows aktiviert wurde. Davon abhängig werden Meldungen an einen Benutzer geschickt.

```
:SET &TNAME# = SYS_ACT_TOP_NAME()
:SET &PNAME# = SYS_ACT_PARENT_NAME()

:IF &TNAME# = &PNAME#
:   IF &TNAME# = " "
:      SEND_MSG ADMIN,AUTOMIC,"Aufgabe läuft nicht in einem Workflow."
:   ELSE
:      SEND_MSG ADMIN,AUTOMIC,"Aufgabe läuft im Workflow &TNAME#.
Keine übergeordneten Workflows."
:   ENDIF
:ELSE
:   SEND_MSG ADMIN,AUTOMIC,"Name des obersten Workflows: &TNAME#."
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_ACT_TOP_NR	Ermittelt die laufende Nummer (RunID) des obersten Workflows.

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7.21 SYS_ACT_TOP_NR

Script-Funktion: Ermittelt die laufende Nummer (RunID) des obersten Workflows.

Syntax

SYS_ACT_TOP_NR*()*

Rückgabewerte
RunID des obersten Workflows " " - Die Aufgabe wurde nicht durch einen Workflow aktiviert.

Beispiel

Im Beispiel werden die Rückgabewerte der Script-Funktionen verglichen um festzustellen, ob die Aufgabe ohne Workflow innerhalb eines Workflows oder im Rahmen verschachtelter Workflows aktiviert wurde. Davon abhängig werden Meldungen an einen Benutzer geschickt.

```
:SET &TNR# = SYS_ACT_TOP_NR()
:SET &PNR# = SYS_ACT_PARENT_NR()

:IF &TNR#    = &PNR#
:   IF &TNR# = " "
:      SEND_MSG ADMIN,AUTOMIC,"Aufgabe läuft nicht in einem Workflow."
:   ELSE
:      SEND_MSG ADMIN,AUTOMIC,"Aufgabe läuft im Workflow mit RunID
&TNR#. Keine übergeordneten Workflows."
:   ENDIF
:ELSE
:   SEND_MSG ADMIN,AUTOMIC,"RunID des obersten Workflows: &TNR#."
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_ACT_TOP_NAME	Liefert den Namen des obersten Workflows.

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7.22 SYS_ACT_USERID

Script-Funktion: Liefert die Benutzerkennung, unter welcher der Job laufen wird.

Syntax

SYS_ACT_USERID*()*

Rückgabewert

Benutzername aus dem Login-Objekt.

Anmerkungen

Diese Script-Funktion kann nur in Jobs verwendet werden. Sie ermittelt die Benutzerkennung, mit der ein Job ausgeführt wird. Sie wird aus den Login-Informationen des Login-Objektes gelesen, welches in der Registerkarte "Attribute" für den Job definiert wurde.

Die Script-Funktion liefert das gleiche Ergebnis wie GET_ATT mit Attribut USERID.

Beispiel

Im Beispiel werden Benutzerkennung und komplette Login-Information im Aktivierungsprotokoll ausgegeben.

```
:SET  &URET# = SYS_ACT_USERID()
 :SET  &LRET# = GET_ATT(LOGIN_INFO)
 :PRINT  "Benutzerkennung: &URET#", "gesamte Login-Information:
&LRET#"
```

Siehe auch:

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7.23 SYS_LAST_RESTART_POINT

Script-Funktion: Liefert die Bezeichnung des vorangegangenen Wiederanlauf-Punktes im Script.

Syntax

SYS_LAST_RESTART_POINT*()*

Rückgabewerte

Name des zuletzt durchlaufenen Wiederanlauf-Punktes.
" " - Es wurden keine Wiederanlauf-Punkte definiert.

Anmerkungen

Wiederanlauf-Punkte werden mit dem Script-Sprachmittel :RESTART gesetzt.

Beispiel

Im Beispiel werden Dateien aus dem AE-Verzeichnis gesichert. Als vorangegangener Wiederanlauf-Punkt wird "BIN" ins Aktivierungsprotokoll geschrieben.

```
:RESTART BIN, "Programmverzeichnis"
COPY C:\AUTOMICG\BIN\*.* C:\BACKUP\AUTOMICG\BIN\*.* /Y
:INC DOS.ERRORLEVEL

:SET &RET# = SYS_LAST_RESTART_POINT()
:PRINT "Letzter Wiederanlaufpunkt: &RET#"

:RESTART DB, "Datenbank-Verzeichnis"
XCOPY C:\AUTOMICG\DB\*.* C:\BACKUP\AUTOMICG\DB\*.* /E /Y
:INC DOS.ERRORLEVEL
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:RESTART	Setzt Wiederanlauf-Punkte in einem aktivierbaren Objekt.
SYS_LAST_RESTART_TEXT	Liefert den Text des vorangegangenen Wiederanlauf-Punktes im Script.

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7.24 SYS_LAST_RESTART_TEXT

Script-Funktion: Liefert den Text des vorangegangenen Wiederanlauf-Punktes im Script.

Syntax

SYS_LAST_RESTART_TEXT*()*

Rückgabewerte
Text des zuletzt durchlaufenen Wiederanlauf-Punktes. "Restart Point *Name des Wiederanlauf-Punktes*" - Es wurde ein Wiederanlauf-Punkt ohne Text definiert. " " - Es wurden keine Wiederanlauf-Punkte definiert.

Anmerkungen

Wiederanlauf-Punkte werden mit dem Script-Sprachmittel :RESTART gesetzt.

Beispiele

Im ersten Beispiel werden Dateien aus dem AUTOMIC-Verzeichnis gesichert. Als Text des vorangegangenen Wiederanlauf-Punktes wird "Programmverzeichnis" ins Aktivierungsprotokoll geschrieben.

```
:RESTART BIN, "Programmverzeichnis"
COPY C:\AUTOMICG\BIN\*.* C:\BACKUP\AUTOMICG\BIN\*.* /Y
:INC DOS.ERRORLEVEL

:SET &RET# = SYS_LAST_RESTART_TEXT()
:PRINT "Letzter Wiederanlaufpunkt: &RET#"

:RESTART DB, "Datenbank-Verzeichnis"
XCOPY C:\AUTOMICG\DB\*.* C:\BACKUP\AUTOMICG\DB\*.* /E /Y
:INC DOS.ERRORLEVEL
```

Das zweite Beispiel liefert das Ergebnis "Restart Point BIN", da kein Text für den Wiederanlauf-Punkt festgelegt wurde.

```
:RESTART BIN
:SET  &RET# = SYS_LAST_RESTART_TEXT()
:PRINT  &RET#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:RESTART	Setzt Wiederanlauf-Punkte in einem aktivierbaren Objekt.
SYS_LAST_RESTART_POINT	Liefert die Bezeichnung des vorangegangenen Wiederanlauf-Punktes im Script.

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.7.25 SYS_RESTART_POINT

Script-Funktion: Liefert den Wiederanlauf-Punkt, mit dem das Objekt ausgeführt wird.

Syntax

SYS_RESTART_POINT*()*

Rückgabewerte
Name des Wiederanlauf-Punktes " " - Das Script wird nicht ab einem bestimmten Wiederanlauf-Punkt oder generell nicht im Wiederanlauf-Modus durchgeführt.

Anmerkungen

Die Script-Funktion ermittelt, ab welchem Wiederanlauf-Punkt das Script prozessiert wird, wenn eine Aufgabe mit *Ausführen...* unter der Angabe eines Wiederanlauf-Punktes gestartet wurde.

Wiederanlauf-Punkte werden mit dem Script-Sprachmittel :RESTART gesetzt.

Beispiel

Im Beispiel werden Dateien aus dem AUTOMIC-Verzeichnis gesichert. Bei einem Wiederanlauf des Jobs unter Angabe von "DB" als Wiederanlauf-Punkt werden nur die Dateien aus dem Datenbank-Verzeichnis (einschließlich aller Unterverzeichnisse) kopiert. Ein entsprechender Hinweis wird im Aktivierungsprotokoll ausgegeben.

```
:RESTART BIN, "Programmverzeichnis"
COPY C:\AUTOMICG\BIN\*.* C:\BACKUP\AUTOMICG\BIN\*.* /Y
:INC DOS.ERRORLEVEL

:RESTART DB, "Datenbank-Verzeichnis"
XCOPY C:\AUTOMICG\DB\*.* C:\BACKUP\AUTOMICG\DB\*.* /E /Y
:INC DOS.ERRORLEVEL

:SET &RET# = SYS_RESTART_POINT()
:PRINT "Script wurde ab &RET# prozessiert."
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:RESTART	Setzt Wiederanlauf-Punkte in einem aktivierbaren Objekt.
RESTART_UC_OBJECT	Wiederholt die Durchführung einer Aufgabe.
SYS_ACT_RESTART	Ermittelt, ob Objekt im Wiederanlauf-Modus aktiviert wurde.
SYS_ACT_RESTART_ME_NR	Liefert die laufende Nummer (RunID) des im Wiederanlauf-Modus aktivierten Objektes.
SYS_LAST_RESTART_POINT	Liefert die Bezeichnung des vorangegangenen Wiederanlauf-Punktes im Script.
SYS_LAST_RESTART_TEXT	Liefert den Text des vorangegangenen Wiederanlauf-Punktes im Script.

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.8 Benutzerdaten

3.8.1 IS_GROUP_MEMBER

Script-Funktion: Prüft, ob ein Benutzer Mitglied in einer BenutzerGruppe ist.

Syntax

IS_GROUP_MEMBER *(Benutzer, BenutzerGruppe)*

Syntaxteil	Beschreibung/Format
Benutzer	Name eines Benutzers-Objektes. Format: Script-Literal oder Script-Variable
BenutzerGruppe	Name einer BenutzerGruppen-Objektes. Format: Script-Literal oder Script-Variable

Rückgabewerte
"1" - Der Benutzer ist Mitglied der BenutzerGruppe. "0" - Der Benutzer ist kein Mitglied der BenutzerGruppe bzw. der Benutzer und/oder die BenutzerGruppe existieren nicht.

Anmerkungen

Mit der Script-Funktion kann die Mitgliedschaft eines Benutzers in einer BenutzerGruppe überprüft werden. Beide müssen sich im dem Mandanten befinden, in dem die Script-Funktion aufgerufen wird.

Beispiel

Im Beispiel wird geprüft, ob der Benutzer "BU/UC4" Mitglied in der BenutzerGruppe "ADMIN" ist. Das Ergebnis wird im Aktivierungsprotokoll ausgegeben.

```
:SET &USGR#    = "ADMIN"
 :SET &MEMBER# = IS_GROUP_MEMBER("BU/UC4", &USGR#)
 :IF &MEMBER# = 1
:    PRINT "Mitglied in &USGR#"
 :ELSE
:    PRINT "Kein Mitglied in &USGR#"
 :ENDIF
```

Siehe auch:

Script-Sprachmittel - Benutzerdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.8.2 SYS_ACT_CLIENT, SYS_USER_CLIENT

Script-Funktion: Liefert die Nummer des aktuellen Mandanten.

Syntax

SYS_ACT_CLIENT *()*
SYS_USER_CLIENT *()*

Rückgabewert
Aktuelle Mandantennummer

Anmerkungen

Die zurückgegebene Zahl ist die 4-stellige Mandantennummer mit führenden Nullen. Als Mandant wird jener herangezogen in dem die Aufgabe läuft, die die Script-Funktion aufruft.

Beispiel

Die Nummer des aktuellen Mandanten wird in eine Script-Variable gelesen. Eine Zeile mit dieser Information (z.B: "0003") wird im Report ausgegeben.

```
:SET &MANDANT# = SYS_ACT_CLIENT()
 :PRINT "Aktueller Mandant: &MANDANT#"
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_ACT_CLIENT_TEXT	Liefert den Text des aktuellen Mandanten.

Script-Sprachmittel - Benutzerdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.8.3 SYS_ACT_CLIENT_TEXT

Script-Funktion: Liefert den Text des aktuellen Mandanten.

Syntax

SYS_ACT_CLIENT_TEXT *()*

Rückgabewerte
Titel des aktuellen Mandanten. " " - Ein Titel wurde nicht definiert.

Anmerkungen

Die Script-Funktion gibt den Titel des Mandanten zurück. Dieser kann in der Registerkarte **Kopf** des Mandanten-Objektes eingetragen werden. Als Mandant wird jener herangezogen in dem die Aufgabe läuft, die die Script-Funktion aufruft.

Beispiel

Der Text des aktuellen Mandanten wird in eine Script-Variable gelesen. Das Ergebnis "Aktueller Mandantitel: Produktivsystem" beispielsweise wird im Report ausgegeben.

```
:SET &TXT#=SYS_ACT_CLIENT_TEXT()
 :PRINT "Aktueller Mandantentitel: &TXT#"
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_ACT_CLIENT oder SYS_USER_CLIENT	Liefert die Nummer des aktuellen Mandanten.

Script-Sprachmittel - Benutzerdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.8.4 SYS_USER_ALIVE

Script-Funktion: Prüft, ob ein Benutzer mit einem UserInterface an AE angemeldet ist.

Syntax

SYS_USER_ALIVE*(Name, Abteilung)*

Syntaxteil	Beschreibung/Format
Name	Name des Benutzers. Format: Script-Literal oder Script-Variable
Abteilung	Abteilung des Benutzers. Format: Script-Literal oder Script-Variable

Rückgabewerte
"Y" - Der Benutzer ist am UserInterface angemeldet. "N" - Der Benutzer ist nicht am UserInterface angemeldet bzw. das angegebene Benutzer-Objekt existiert nicht.

Anmerkungen

Die Script-Funktion prüft, ob ein Benutzer an AE angemeldet ist.

Der Benutzer wird durch *Name* und *Abteilung* benannt (z.B. MEIER/DEV). Es können nur Benutzer aus dem eigenen Mandanten geprüft werden.

⚠ Benutzer, die über Dienstprogramme oder CallAPI an AE angemeldet sind, werden nicht erkannt.

Die Script-Funktion gibt "N" zurück, wenn der Benutzer nicht angemeldet ist oder wenn das Benutzer-Objekt nicht im aktuellen Mandanten existiert. In letzterem Fall können Sie mit der Script-Anweisung :ON_ERROR auf diesen Fehler reagieren und mit den Script-Funktionen zur Fehlerbehandlung analysieren. Das Script wird je nach Einstellung entweder weiter prozessiert oder abgebrochen.

Beispiel

Im Beispiel wird geprüft, ob ein Benutzer an AE angemeldet ist. Name und Abteilung werden der Script-Funktion als Script-Variablen übergeben. Das Script bricht ab, wenn der Benutzer "MEIER/DEV" nicht definiert ist. Das Ergebnis wird im Aktivierungsprotokoll ausgegeben.

```
:ON_ERROR ABEND
:SET &USR# = "MEIER"
:SET &DEP# = "DEV"
:SET &RET# = SYS_USER_ALIVE(&USR#, &DEP#)
:PRINT &RET#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_USER_DEP	Liefert die Abteilung des Benutzers, der die Aufgabe startete.
SYS_USER_ LANGUAGE	Liefert die Sprache in der der Server die Logdateien erzeugt.
SYS_USER_LNAME	Liefert den Vor- und Nachnamen des Benutzers, der die Aufgabe startete.
SYS_USER_NAME	Liefert den Namen des Benutzers, der die Aufgabe startete.

Script-Sprachmittel - Benutzerdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.8.5 SYS_USER_DEP

Script-Funktion: Liefert die Abteilung des Benutzers, der die Aufgabe startete.

Syntax

SYS_USER_DEP*()*

Rückgabewert
Abteilung des Benutzers

Anmerkungen

Die Script-Funktion ermittelt die Abteilung des Benutzers (z.B: "AUTOMIC" bei "MEIER/AUTOMIC"), der die Aufgabe gestartet hat.

Beachten Sie, dass Aufgaben sowohl manuell, per Script-Sprachmittel "ACTIVATE_UC_OBJECT", über Workflows und Schedules gestartet werden können. Ist daher die Aufgabe in der "SYS_USER_DEP" aufgerufen, wird beispielsweise Teil eines Schedules, so liefert die Script-Funktion die Abteilung des Benutzers der den Schedule gestartet hat.

Der Benutzer ist auch in der Statistikübersicht erkennbar.

Beispiel

Im Beispiel wird die mit der Script-Funktion ermittelte Abteilung aus der Anmeldung in das Aktivierungsprotokoll geschrieben.

```
:SET  &LOGDEP# = SYS_USER_DEP()
:PRINT  "Anmelde-Information: Abteilung des Benutzers ist &LOGDEP#."
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_USER_ALIVE	Prüft, ob der Benutzer mit einem UserInterface an AE angemeldet ist.
SYS_USER_LANGUAGE	Liefert die Sprache in der der Server die Logdateien erzeugt.
SYS_USER_LNAME	Liefert den Vor- und Nachnamen des Benutzers, der die Aufgabe startete.
SYS_USER_NAME	Liefert den Namen des Benutzers, der die Aufgabe startete.

Script-Sprachmittel - Benutzerdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.8.6 SYS_USER_LNAME

Script-Funktion: Liefert den Vor- und Nachnamen des Benutzers, der die Aufgabe startete.

Syntax

SYS_USER_LNAME*()*

Rückgabewerte
Vor- und Nachname des Benutzers. " " - Der Name wurde nicht definiert.

Anmerkungen

Die Script-Funktion ermittelt den Vor- und Nachnamen des Benutzers (z.B: "Hans Meier"), der die Aufgabe gestartet hat. Der vollständige Name wird in der Registerkarte **Benutzer** des Benutzer-Objektes definiert.

Beachten Sie, dass Aufgaben sowohl manuell, per Script-Sprachmittel "ACTIVATE_UC_OBJECT", über Workflows und Schedules gestartet werden können. Ist daher die Aufgabe in der "SYS_USER_LNAME" aufgerufen wird beispielsweise Teil eines Schedules, so liefert die Script-Funktion den Namen des Benutzers der den Schedule gestartet hat.

Der Benutzer ist auch in der Statistikübersicht erkennbar.

Beispiel

Im Beispiel wird der mit der Script-Funktion ermittelte Vor- und Nachname aus der Anmeldung in das Aktivierungsprotokoll geschrieben.

```
:SET &LOGLN# = SYS_USER_LNAME()
:PRINT "Anmelde-Information: Vor-und Nachname des Benutzers ist
&LOGLN#."
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_USER_ALIVE	Prüft, ob der Benutzer mit einem UserInterface an AE angemeldet ist.
SYS_USER_DEP	Liefert die Abteilung des Benutzers, der die Aufgabe startete.
SYS_USER_LANGUAGE	Liefert die Sprache in der der Server die Logdateien erzeugt.
SYS_USER_NAME	Liefert den Namen des Benutzers, der die Aufgabe startete.

Script-Sprachmittel - Benutzerdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.8.7 SYS_USER_NAME

Script-Funktion: Liefert den Namen des Benutzers, der die Aufgabe startete.

Syntax

SYS_USER_NAME*()*

Rückgabewert
Name des Benutzers.

Anmerkungen

Die Script-Funktion ermittelt den Namen des Benutzers (z.B: "MEIER" bei "MEIER/DEV"), der die Aufgabe gestartet hat.

Beachten Sie, dass Aufgaben sowohl manuell, per Script-Sprachmittel "ACTIVATE_UC_OBJECT", über Workflows und Schedules gestartet werden können. Ist daher die Aufgabe in der "SYS_USER_NAME" aufgerufen wird beispielsweise Teil eines Schedules, so liefert die Script-Funktion den Namen des Benutzers der den Schedule gestartet hat.

Der Benutzer ist auch in der Statistikübersicht erkennbar.

Beispiel

Im Beispiel wird der mit der Script-Funktion ermittelte Name aus der Anmeldung in das Aktivierungsprotokoll geschrieben.

```
:SET &LOGN# = SYS_USER_NAME()
:PRINT "Anmelde-Information: Name des Benutzers ist &LOGN#."
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_USER_ALIVE	Prüft, ob der Benutzer mit einem UserInterface an AE angemeldet ist.
SYS_USER_DEP	Liefert die Abteilung des Benutzers, der die Aufgabe startete.
SYS_USER_LANGUAGE	Liefert die Sprache in der der Server die Logdateien erzeugt.
SYS_USER_LNAME	Liefert den Vor- und Nachnamen des Benutzers, der die Aufgabe startete.

Script-Sprachmittel - Benutzerdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.9 Datensequenzen

3.9.1 :CLOSE_PROCESS

Script-Anweisung: Verwirft eine Datensequenz innerhalb eines Scripts.

Syntax

:CLOSE_PROCESS *Datensequenz-Referenz*

Syntaxteil	Beschreibung/Format
Datensequenz-Referenz	Script-Variable, die eine Datensequenz-Referenz enthält. Format: Script-Variable

Anmerkungen

Die Script-Anweisung :CLOSE_PROCESS verwirft eine nicht mehr benötigte Datensequenz. Damit wird besonders bei verschachtelter Verarbeitung von Datensequenzen in einem Script Arbeitsspeicher freigegeben.

Datensequenzen werden von den folgenden Script-Sprachmitteln bereitgestellt:

- PREP_PROCESS
- PREP_PROCESS_AGENTGROUP
- PREP_PROCESS_COMMENTS
- PREP_PROCESS_FILE
- PREP_PROCESS_FILENAME
- PREP_PROCESS_REPORT
- PREP_PROCESS_VAR

Script-Variablen, die eine Referenz auf eine Datensequenzen enthalten, können keine neuen Werte zugewiesen werden. Erst wenn Sie diese mit :CLOSE_PROCESS freigeben, können Sie sie erneut verwenden.

Nach dem Aufruf :CLOSE_PROCESS enthält die Variable keinen Wert.

Beispiel

Das Beispiel verwirft die Datensequenz, deren Referenz in der Script-Variablen "&HND#" gespeichert ist.

```
:CLOSE_PROCESS   &HND#
```

Siehe auch:

Script-Sprachmittel - Datensequenzen
Beispielsammlung
Endestatus aufgrund des Reportinhaltes setzen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.9.2 :PROCESS... :TERM_PROCESS... :ENDPROCESS

Script-Anweisungen: Definition einer Schleife zur zeilenweisen Bearbeitung einer Datensequenz.

Syntax

:PROCESS ***Datensequenz-Referenz***
[Anweisungen]
:TERM_PROCESS
:ENDPROCESS

Syntaxteil	Beschreibung/Format

:PROCESS	Beginn der Prozess-Schleife.
Datensequenz-Referenz	Referenz auf die zu verarbeitende Datensequenz. Format: Script-Variable
Anweisungen	Eine oder mehrere Anweisungen, die bei jedem Schleifendurchlauf ausgeführt werden. Format: Script-Anweisung
:TERM_PROCESS	Anweisung zum Verlassen der Prozess-Schleife.
:ENDPROCESS	Ende der Prozess-Schleife.

Anmerkungen

Die Script-Anweisungen :PROCESS und :ENDPROCESS erlauben eine zeilenweise Verarbeitung von Datensequenzen. Diese werden von den folgenden Script-Sprachmitteln bereitgestellt:

- PREP_PROCESS
- PREP_PROCESS_AGENTGROUP
- PREP_PROCESS_COMMENTS
- PREP_PROCESS_FILE
- PREP_PROCESS_FILENAME
- PREP_PROCESS_REPORT
- PREP_PROCESS_VAR

Mit jedem Schleifendurchlauf wird eine neue Zeile der Datensequenz gelesen. Das wird solange wiederholt, bis deren Ende erreicht ist bzw. die Schleife explizit mit :TERM_PROCESS beendet wird.

Der Zeileninhalt kann mit der Script-Funktion GET_PROCESS_LINE ermittelt werden.

Eine leere Datensequenz führt zu keiner Fehlermeldung. Die Bearbeitung, die zwischen :PROCESS und :ENDPROCESS definiert ist, wird einfach nicht durchlaufen.

Beispiel

Im folgenden Beispiel werden die Verzeichnisse eines Laufwerkes ermittelt und innerhalb der Prozess-Schleife mit der :PRINT-Anweisung in das Aktivierungsprotokoll ausgegeben.

```
:SET &HND# = PREP_PROCESS("PC01","WINCMD","*DIR*","CMD=DIR C:")
:PROCESS &HND#
: SET &LINE# = GET_PROCESS_LINE(&HND#)
: PRINT &LINE#
:ENDPROCESS
```

Siehe auch:

Script-Sprachmittel - Datensequenzen
Beispielsammlung
Endestatus aufgrund des Reportinhaltes setzen
Aufruf eines MBeans
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.9.3 CREATE_PROCESS

Script-Funktion: Erstellt ein neue Datensequenz.

Syntax

CREATE_PROCESS*(Modus *[[, Datensequenz-Referenz1] , Datensequenz-Referenz2]**)***

Syntaxteil	Beschreibung/Format
Modus	Modus, wie die Datensequenz erstellt werden soll Format: AE-Name Erlaubte Werte: • NEW - Neue leere Datensequenz erstellen • JOIN - Zwei Datensequenzen zu einer zusammenfügen • DUPLICATE - Kopie einer bestehenden Datensequenz anlegen
Datensequenz-Referenz1	Referenz auf die Datensequenz, die entweder dupliziert oder kombiniert werden soll (abhängig vom Modus). Format: Script-Variable Dieser Parameter ist nur beim *Modus* JOIN oder DUPLICATE anzugeben.
Datensequenz-Referenz2	Referenz auf die Datensequenz, welche mit Datensequenz1 kombiniert werden soll. Format: Script-Variable Dieser Parameter ist nur beim *Modus* JOIN anzugeben.

Rückgabewert
Referenz auf die neu erstellte Datensequenz

Anmerkungen

Das Script-Sprachmittel erstellt eine neue Datensequenz und liefert als Rückgabewert die Referenz darauf zurück. Mit dem Parameter *Modus* wird bestimmt, ob diese leer angelegt oder mit Zeilen von anderen Datensequenzen befüllt wird.

Datensequenzen können auch direkt mit den PREP_PROCESS* - Sprachmitteln erstellt werden.

Mit dem Sprachmittel PUT_PROCESS_LINE haben Sie die Möglichkeit, der Datensequenz ein oder mehrere Zeilen hinzuzufügen.

Anschließend können Sie die erstellte Datensequenz mittels GET_PROCESS_LINE auslesen. Um dabei mehrere oder alle Zeilen möglichst einfach abzuarbeiten, können :PROCESS-Schleifen verwendet werden.

Beispiele

Im ersten Beispiel wird eine neue, leere Datensequenz erstellt und dieser anschließend 2 Zeilen mit jeweils 3 Spalten hinzugefügt.

```
:SET &HND# = CREATE_PROCESS(NEW)
:SET &LINE1# = "Test1,Test2,Test3"
:SET &RET# = PUT_PROCESS_LINE(&HND#, &LINE1#, ",")
:DEFINE &LINE2#, string, 3
:FILL &LINE2#[] = GET_VAR(TEST.VAR, KEY1)
:SET &RET# = PUT_PROCESS_LINE(&HND#, &LINE2#)
```

Das zweite Beispiel zeigt das Duplizieren einer Datensequenz.

```
:SET &HND1# = PREP_PROCESS_VAR(VARA.DB,"*WIN*")
:SET &HND2# = CREATE_PROCESS(DUPLICATE,&HND1#)
```

Im letzten Beispiel sehen Sie wie 2 Datensequenzen zu einer kombinert werden.

```
:SET &HND1# = PREP_PROCESS_VAR(VARA.DB1,"*WIN*")
:SET &HND2# = PREP_PROCESS_VAR(VARA.DB2,,"*JOBS*",1)
:SET &HND# = CREATE_PROCESS(JOIN,&HND1#,&HND2#)
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:CLOSE_PROCESS	Verwirft eine Datensequenz innerhalb eines Scripts.
:PROCESS... :TERM_ PROCESS... :ENDPROCESS	Definition einer Schleife zur zeilenweisen Bearbeitung einer Datensequenz, wie beispielsweise den Inhalt einer sequentiellen Datei oder das textmäßige Ergebnis eines Kommandos.
GET_PROCESS_ LINE	Ermittelt den aktuellen Zeileninhalt einer Datensequenz.
PUT_PROCESS_ LINE	Fügt einer bestimmten Datensequenz eine Zeile hinzu.

Script-Sprachmittel - Datensequenzen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.9.4 GET_PROCESS_INFO

Script-Funktion: Informationen einer Datensequenz ermitteln.

Syntax

GET_PROCESS_INFO*(Datensequenz-Referenz, Information [, Zeile])*

Syntaxteil	Beschreibung/Format

Datensequenz-Referenz	Referenz auf die Datensequenz, deren Informationen gelesen werden soll. Format: Script-Variable
Information	Angabe, welche Information ermittelt wird. Format: AE-Name Erlaubte Werte: • INDEX - Aktuelle Position in der Datensequenz ermitteln. Nur sinnvoll, innerhalb von Prozess-Schleifen. • COLUMNS - Anzahl der Spalten der angegebenen *Zeile*. Wird keine Zeile angegeben, so wird die Spaltenanzahl der aktuellen (bei Prozess-Schleifen) bzw. der ersten Zeile ermittelt. • ROWS - Anzahl der Zeilen der Datensequenz.
Zeile	Zeile, deren Spalten-Anzahl ermittelt werden soll. Format: Zahl ohne Hochkommata, Script-Literal oder Script-Variable Dieser Parameter kann nur angegeben werden, wenn die Anzahl der Spalten ausgelesen wird (*Information* = COLUMNS).

Rückgabewert
Entsprechende Information der Datensequenz

Anmerkungen

Mit der Script-Funktion haben Sie die Möglichkeit, allgemeine Informationen (wie z.B.: die Zeilen- oder Spalten-Anzahl) von Datensequenzen zu ermitteln. Anzugeben ist dabei die Referenz auf die Datensequenz und die zu bestimmende Information.

Soll die Spaltenanzahl ausgelesen werden, so können Sie zusätzlich die Nummer der *Zeile* spezifizieren, die dafür verwendet werden soll. Erfolgt keine Angabe, so wird abhängig von der Verwendung des Sprachmittels folgende Zeile herangezogen:

- Das Sprachmittel wird innerhalb einer Prozess-Schleife verwendet: Zeile, auf welche die Schleife aktuell zugreift.
- Sprachmittel befindet sich außerhalb einer Prozess-Schleife: Erste Zeile

Ist für *Information* der Wert INDEX gesetzt, so ist das Sprachmittel innerhalb einer Prozess-Schleife zu verwenden. Ermittelt wird in diesem Fall die Zeile der Datensequenz, die aktuell von der Schleife bearbeitet wird.

Das Auslesen der Zeilenanzahl von Datensequenz unterliegt keiner besonderen Abhängigkeiten.

⚠ Achten Sie darauf, dass die angegebene Datensequenz existiert und nicht bereits mit dem Sprachmittel :CLOSE_PROCESS geschlossen wurde. Ansonsten kommt es zu einem Laufzeitfehler.

Beispiele

Im folgendem Beispiel wird eine Datensequenz erstellt und mit mehreren Einträgen eines Variablen-Objektes befüllt. Anschließend wird die Anzahl der Spalten pro Zeile bestimmt und im Aktivierungsprotokoll ausgegeben.

```
:SET &HND# = PREP_PROCESS_VAR(VARA.SQL,"*WIN*")
:SET &LNR# = GET_PROCESS_INFO(&HND#, ROWS)

:PROCESS &HND#
:SET &IND# = GET_PROCESS_INFO(&HND#,INDEX)
:SET &COL# = GET_PROCESS_INFO(&HND#,COLUMNS,&IND#)
:PRINT "Zeile &IND# / &LNR# besitzt &COL# Spalten"
:ENDPROCESS

:CLOSE_PROCESS &HND#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:CLOSE_PROCESS	Verwirft eine Datensequenz innerhalb eines Scripts.
:PROCESS... :TERM_ PROCESS... :ENDPROCESS	Definition einer Schleife zur zeilenweisen Bearbeitung einer Datensequenz, wie beispielsweise den Inhalt einer sequentiellen Datei oder das textmäßige Ergebnis eines Kommandos.
CREATE_ PROCESS	Erstellt ein neue Datensequenz.
GET_PROCESS_ LINE	Ermittelt den aktuellen Zeileninhalt einer Datensequenz.

Script-Sprachmittel - Datensequenzen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.9.5 GET_PROCESS_LINE

Script-Funktion: Ermittelt den aktuellen Zeileninhalt einer Datensequenz.

Syntax

GET_PROCESS_LINE*(Datensequenz-Referenz* *[, [Spalte] [, [***STR_SUBSTITUTE_VAR***] [, Zeile]]] ***)***
GET_PROCESS_LINE*(Datensequenz-Referenz* *[, [Spalte] [, [***STR_SUB***] [, Zeile]] ***)***

Syntaxteil	Beschreibung/Format
Datensequenz-Referenz	Referenz auf die zu verarbeitende Datensequenz. Format: Script-Variable

Spalte	Zugriff auf eine Spalte der Datensequenz-Zeile. • Datensequenz, deren Zeilen in Spalten unterteilt wurde. Format: Zahl ohne Hochkommata, Script-Literal oder Script-Variable Erlaubte Werte: "1" bis "22" oder Spaltenname • Datensequenz eines Variablen-Objektes (PREP_PROCESS_VAR) Format: Zahl ohne Hochkommata oder Script-Variable Erlaubte Werte: "1", "2" bis n "1" = Key bzw. Result-Spalte. Werte-Spalte von Variablen-Objekten des Typs "Dateiliste". "2" - "6" = Wertespalten von statischen Variablen-Objekten "2" bis n = Nummer der Wertespalten von Variablen-Objekten. Dynamische Variablen (SQL, SQLI und Multi) können beliebig viele Spalten enthalten. Dies richtet sich nach der jeweiligen Datenquelle. Statische Variablen besitzen nur 5 Wertespalten ("2" bis "6"). FILELIST-Variablen besitzen generell nur eine Spalte, die mit "1" anzusprechen ist. • Datensequenz eines Kommentars (PREP_PROCESS_COMMENTS) Format: Zahl ohne Hochkommata oder Script-Variable Erlaubte Werte: "1", "2" und "3" "1" = Zeitstempel "2" = Name des Benutzers "3" = Kommentartext • Datensequenz eines PromptSet-Objektes (PREP_PROCESS_PROMPTSET) Format Zahl ohne Hochkommate, AE-Name, Script-Literal oder Script-Variable Erlaubte Werte: "1" oder "VARIABLE_NAME" = Name der PromptSet-Variable des PromptSet-Elementes (ohne führendes &) "2" oder "CUSTOM_FIELD" = Inhalt des benutzerdefinierten Feldes "3" oder "DEFAULT" = Standardwert des PromptSet-Elementes "4" oder "TYPE" = Typ des PromptSet-Elementes (z.B.: "text" oder "integer") "5" oder "DATA_REFERENCE" = Name des Referenz-Objektes (Variable oder Kalender) "6" oder "DATA_REFERENCE_KEY" = Begriff des Referenz-Kalenderobjektes (wenn vorhanden)

STR_SUBSTITUTE_VAR **STR_SUB_VAR**	Innerhalb des Zeileninhalts der Datensequenz soll zusätzlich nach Script-Variablen gesucht und diese durch ihre Werte ersetzt werden. Wenn Sie diesen Parameter verwenden jedoch keine *Spalte* angeben, ist trotzdem deren Komma zu setzen. Beispiel: `:SET &WERT# = GET_PROCESS_LINE(&HND#,,STR_SUB_VAR)`
Zeile	Zeile der Datensequenz, auf die zugegriffen werden soll. Format: Zahl ohne Hochkommate, Script-Literal oder Script-Variable

Anmerkungen

Die Script-Funktion GET_PROCESS_LINE liest eine Zeile oder Spalten aus einer Datensequenz. Diese wird von folgenden Script-Sprachmitteln bereitgestellt:

- CREATE_PROCESS
- LOAD_PROCESS
- PREP_PROCESS
- PREP_PROCESS_AGENTGROUP
- PREP_PROCESS_COMMENTS
- PREP_PROCESS_DOCU
- PREP_PROCESS_FILE
- PREP_PROCESS_PROMPTSET
- PREP_PROCESS_FILENAME
- PREP_PROCESS_REPORT
- PREP_PROCESS_VAR

Die Script-Funktion GET_PROCESS_LINE wird innerhalb der Script-Anweisungen :PROCESS und :ENDPROCESS verwendet, welche eine Schleife für die Bearbeitung der Datensequenz bilden und mit jedem Schleifendurchlauf eine neue Zeile der Datensequenz zur Verfügung stellen.

Bitte beachten Sie, dass GET_PROCESS_LINE sich ab v9 in einer PROCESS-Schleife befinden muss.

Die einzelnen Zeilen einer Datensequenz können auch in Spalten unterteilt sein. Dies wird schon bei der Erstellung der Datensequenz mit den Parametern COL=LENGTH bzw. COL=DELIMITER definiert. Der Zugriff auf eine einzelne Spalte erfolgt durch deren Nummer (1 - erste Spalte) oder sofern festgelegt über den Spaltennamen.

Geben Sie die Spalte nicht an oder verwenden Sie "0", so wird die gesamte Zeile zurückgegeben, daher alle Spalten.

Wenn die Werte eines Variablen-Objektes ermittelt werden, enthält die aktuelle Zeile den Key (statische Variable) bzw. die Result-Spalte (dynamische Variable) plus die zugehörigen Werte. Mit dem Wert "1" für *Spalte* wird der Key/Result, mit "2" bis "6" (statische Variablen) die jeweilige Wertspalte gelesen. Ist im Variablen-Objekt (SQL, SQLI, MULTI) kein Result-Format definiert, so entspricht die Result-Spalte der ersten Werte-Spalte.
Die Anzahl der Wertespalten von dynamischen Variablen mit der Quelle **SQL**, "SQL - intern" oder "Multi" ist nicht begrenzt und richtet sich nach den Einstellungen im Variablen-Objekt bzw. nach der Datenquelle. Auch die Spalten von Kommentaren können auf diese Weise einzeln angesprochen werden.

Wird die Spaltennummer bei der Datensequenz von Variablen-Objekten nicht angegeben, so wird der Wert aller Spalten (inklusive Key/Result) getrennt durch die Zeichen "§§§" zurückgeliefert.

Für den Zugriff auf SAP-Monitore gilt ein besonderes Verfahren. Der SAP-Agent teilt die einzelnen Zeilen, welche ein SAP-Monitor liefert, in feste Spalten auf. Er hinterlegt Spaltennamen und Spaltenbreiten als erste Zeile in der Datei, die durch die Automation Engine verarbeitet wird. Mit GET_PROCESS_LINE kann auf folgende Spalten zugegriffen werden:

- CONTEXT - Name des Monitoring-Kontextes
- PATH - Pfadangabe eines Wertes,
- NAME -Name des Wertes,
- VALUE - Aktueller Wert,
- STATUS - Status: 1 = grün, 2 = gelb, 3 = rot,
- DATE - Datum der Prüfung,
- TIME - Zeitpunkt der Prüfung.

Sie können zusätzlich den Parameter STR_SUBSTITUTE_VAR angeben, wobei auch die Kurzform STR_SUB_VAR gültig ist. Dies hat zur Folge, dass innerhalb der Zeile Script-Variablen gesucht und diese durch ihre Werte ersetzt werden. Besonders hilfreich ist die Funktion in Verbindung mit PREP_PROCESS_FILE. Bereiten Sie in einer Datei einen Text vor, der an veränderbaren Stellen Script-Variablen enthält, um diesen leicht weiterverarbeiten zu können. Doch egal welcher Quelle die Datensequenz entstammt (Datei, Report, Variablen-Objekt,...), die Werte selbst sind im Script zu setzen (siehe Beispiel 5 unten).

Weiters ist der Zugriff auf eine bestimmte *Zeile* der Datensequenz möglich. Geben Sie dazu die Nummer der Zeile beim entsprechenden Parameter der Script-Funktion an. Die erste Zeile besitzt die Nummer 1. Zeilennummern von Datensequenzen können auch mit dem Sprachmittel GET_PROCESS_INFO ermittelt werden.

Beachten Sie, dass GET_PROCESS_LINE standardmäßig Leerzeichen am Ende der auszulesenden Zeile abschneidet! Dieses Verhalten kann jedoch der Administrator mit der Einstellung GET_PROCESS_LINE_RTRIM in der Variablen UC_SYSTEM_SETTINGS deaktivieren.

Beispiele

Im ersten Beispiel werden die Verzeichnisse eines Laufwerkes ermittelt und innerhalb der Prozess-Schleife mit der :PRINT-Anweisung in das Aktivierungsprotokoll ausgegeben.

```
:SET  &HND# = PREP_PROCESS("PC01","WINCMD","*DIR*","CMD=DIR C:")
:PROCESS  &HND#
:   SET  &LINE# = GET_PROCESS_LINE(&HND#)
:   PRINT  &LINE#
:ENDPROCESS
```

Im zweiten Beispiel werden die Werte einer Variablen ermittelt und innerhalb der Prozess-Schleife mit der :PRINT-Anweisung in das Aktivierungsprotokoll ausgegeben.

```
:SET  &HND#=PREP_PROCESS_VAR(UC_CLIENT_SETTINGS)
:PROCESS  &HND#
:   SET  &RET1# = GET_PROCESS_LINE(&HND#,1)
:   SET  &RET2# = GET_PROCESS_LINE(&HND#,2)
:   PRINT  "&RET1# &RET2#"
:ENDPROCESS
```

Das dritte Beispiel liest alle Zeilen aus einer Logdatei des UserInterfaces, welche Informationen über die Datenbank liefern. Dabei werden Spaltenbreiten und Spaltennamen vorgegeben. Die relevanten Informationen werden im Aktivierungsprotokoll ausgegeben. Die Zwischenräume (Spalten 2 und 4) werden ignoriert.

```
:SET  &HND# = PREP_PROCESS_FILE(WIN21, "F:\AUTOMIC\DIALOG\TEMP\UCDJ_
LOGG_01.TXT","*DB-INFO*","COL=LENGTH","LENGTH_
TAB='8=DATUM,1,6=ZEIT,7,200=TEXT'")
:PROCESS  &HND#
:   SET  &COL1# = GET_PROCESS_LINE(&HND#,1)
:   SET  &COL2# = GET_PROCESS_LINE(&HND#,3)
:   SET  &COL3# = GET_PROCESS_LINE(&HND#,"TEXT")
:   PRINT  "&COL1# &COL2# &COL3#"
:ENDPROCESS
```

Das vierte Beispiel liest den SAP-Monitor "MON1" aus dem Monitorset "AUTOMIC" aus. Es soll auf die einzelnen Spalten der Monitordaten zugegriffen werden. Die Zeilen der Datensequenz werden im Aktivierungsprotokoll ausgegeben.

```
:SET &HND# = PREP_PROCESS
("T01","R3MONITOR","*","MONSET=AUTOMIC","MONNAM=MON1","COL=FILE","UC_
USER_ID=AUTOMIC","UC_SAPMANDANT=001")
:PROCESS &HND#
:   SET &PATH# = GET_PROCESS_LINE(&HND#,"PATH")
:   SET &NAME# = GET_PROCESS_LINE(&HND#,"NAME")
:   SET &VALUE# = GET_PROCESS_LINE(&HND#,"VALUE")
:   SET &STATUS# = GET_PROCESS_LINE(&HND#,"STATUS")
:   SET &DATE# = GET_PROCESS_LINE(&HND#,"DATE")
:   SET &TIME# = GET_PROCESS_LINE(&HND#,"TIME")
:   PRINT "&PATH# &NAME# &VALUE# &STATUS# &DATE# &TIME#"
:ENDPROCESS
```

Definition der Spaltenbreiten und Spaltennamen in der Datei (erste Zeile), die durch den SAP-Agenten bereitgestellt wird.

```
COL=LENGTH,LENGTH_TAB='74=PATH,25=NAME,5=VALUE,2=STATUS,9=DATE,7=TIME'
```

Im fünften Beispiel werden Zeilen einer Textdatei ausgelesen, um zunächst die darin befindlichen Script-Variablen durch ihren Wert zu ersetzen und die so modifizierten Zeilen dann in das Aktivierungsprotokoll auszugeben.

```
:SET  &NAME# = SYS_ACT_ME_NAME()
:SET  &DATE# = SYS_DATE_PHYSICAL("MM/DD/YYYY")
:SET  &TIME# = SYS_TIME_PHYSICAL("HH:MM")
:SET  &JPNAME# = SYS_ACT_PARENT_NAME()

:SET  &HND# = PREP_PROCESS_FILE ("WIN01","C:\AUTOMIC\REPORT.TXT")
:PROCESS  &HND#
:  SET  &RET# = GET_PROCESS_LINE (&HND#,,STR_SUB_VAR)
:  PRINT  &RET#
:ENDPROCESS
```

Auszug aus der vorgefertigten Textdatei REPORT.TXT:

```
&DATE#/&TIME#

Report for &NAME#:

Activated by ProcessFlow: &JPNAME#
```

Siehe auch:

Script-Sprachmittel - Datensequenzen
Beispielsammlung
Endestatus aufgrund des Reportinhaltes setzen
Aufruf eines MBeans

Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.9.6 LOAD_PROCESS

Script-Funktion: Lädt eine gespeicherte Datensequenz.

Syntax

LOAD_PROCESS*(RunID, Key)*

Syntaxteil	Beschreibung/Format
RunID	RunID der Aufgabe, deren gespeicherte Datensequenz geladen werden soll. Format: Zahl ohne Hochkommata, Script-Literal oder Script-Variable
Key	Interner Name, unter welchem die Datensequenz gespeichert wurde. Format: AE Name, Script-Literal oder Script-Variable
Rückgabewert	
Referenz auf die gespeicherte Datensequenz.	

Anmerkungen

Die Script-Funktion lädt eine Datensequenz, die mit dem Sprachmittel SAVE_PROCESS in der Datenbank gespeichert wurde, und stellt diese der Aufgabe zur Verfügung. Dabei ist die *RunID* der Aktivität, über welche die Datensequenz gespeichert wurde und der Begriff, unter welchem die Datensequenz in der Datenbank abgelegt wurde (*Key*), anzugeben. Bei letzterem handelt es sich um den Rückgabewert der Funktion SAVE_PROCESS.

Gespeicherte Datensequenzen können mehrmals von beliebigen Aufgaben geladen und verwendet werden.

⚠ Beachten Sie, dass auch geladene Datensequenzen mit dem Sprachmittel :CLOSE_PROCESS geschlossen werden müssen.

Beispiel

Ein ausführliches Beispiel finden Sie in der Beschreibung von SAVE_PROCESS.

Siehe auch:

Script-Sprachmittel	Beschreibung
:CLOSE_PROCESS	Verwirft eine Datensequenz innerhalb eines Scripts.

:PROCESS... :TERM_ PROCESS... :ENDPROCESS	Definition einer Schleife zur zeilenweisen Bearbeitung einer Datensequenz, wie beispielsweise den Inhalt einer sequentiellen Datei oder das textmäßige Ergebnis eines Kommandos.
CREATE_ PROCESS	Erstellt ein neue Datensequenz.
GET_PROCESS_ LINE	Ermittelt den aktuellen Zeileninhalt einer Datensequenz.
SAVE_PROCESS	Lädt eine gespeicherte Datensequenz.

Script-Sprachmittel - Datensequenzen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.9.7 PREP_PROCESS

Script-Funktion: Führt mit Hilfe von speziellen Job-Objekten (Ereignis-Jobs), Kommandos auf einem Rechner aus, und liefert die Konsolen-Ausgabe als interne Auflistung (Datensequenz) zurück, die zur Weiterverarbeitung genutzt werden kann.

Syntax

PREP_PROCESS*(Host, EreignisJob,* *[Filter]****,Aktion****[, Spaltentrennung]...[,* **UC_ LOGIN=***Login-Objekt]****)***

Syntaxteil	Beschreibung/Format
Host	Rechner (Name des Agenten), auf dem der Ereignis-Job läuft. Format: Script-Literal oder Script-Variable
EreignisJob	Namensteil des Ereignis-Jobs, der ausgeführt werden soll. Format: Script-Literal oder Script-Variable
Filter	Vorgabe für den Zeileninhalt. Groß- und Kleinschreibung wird nicht beachtet. Format: Script-Literal oder Script-Variable Standardwert: "*"

Aktion	Wertzuweisung für eine Script-Variable des Ereignis-Jobs. Format: Script-Literal oder Script-Variable Standard: "" Die Wertzuweisung hat die eigene Syntax: *Variable=Zuweisung* *Variable* - Name der Script-Variablen aus dem Ereignis-Job, welche die Zuweisung aufnimmt. Die Angabe erfolgt ohne das bei Script-Variablen übliche Zeichen "&". *Zuweisung* - Wert für die Script-Variable. Vorrangig Aktion (Kommando, Programm- oder Dateiaufruf), die im Zielsystem ausgeführt werden soll, aber auch sonstige Wertzuweisung (Variablen des Attribut-Dialogs). ⚠ Für die Abfrage des UNIX-Dateisystems gilt eine spezielle Syntax.

Spaltentrennung

Es kann auch festgelegt werden, dass die Zeilen der Datensequenz in Spalten unterteilt werden sollen. Dafür gilt folgendes Format:
COL=*Definition1[, Definition2]*.
Format: Script-Literal oder Script-Variable

Definition1:

Erlaubte Werte: "NONE" (Standardwert), "FILE", "LENGTH", "DELIMITER"

"NONE" = Keine Unterteilung in Spalten.
"FILE" = Verwendung der Spaltendefinitionen in der ersten Zeile der Datei, die der Agent anlegt, wenn Ereignis-Jobs gestartet werden, die das SAP-JCL Sprachmittel R3_GET_MONITOR ausführen (Bei "R3MONITOR" für den Parameter *EreignisJob* wird der vorgefertigte Ereignis-Job "EVENT.R3MONITOR" des Systemmandanten verwendet).
"LENGTH" = Vorgegebene Spaltenbreiten. Erfordert LENGTH_TAB= als *Definition2*.
"DELIMITER" = Spalten werden durch Delimiter getrennt. Erfordert DELIMITER= als *Definition2*.

Definition2:

Legt Spaltenbreiten und Spaltennamen (optional) oder das Begrenzungszeichen fest.
Format: Script-Literal oder Script-Variable

Erlaubte Werte: "LENGTH_TAB" und "DELIMITER"

- "LENGTH_TAB"

  Spaltenbreiten und Spaltennamen (optional) werden in der Form *Spaltenbreite*=[*Spaltenname*] angegeben. Die Spaltenbreite wird über eine Zeichenanzahl definiert. Die einzelnen Spaltendefinitionen sind durch Kommata zu trennen (maximal 22 Spalten). Vor der ersten und nach der letzten Spaltendefinition müssen zusätzlich Hochkommata stehen. Werden für *Definition2* als Script-Literal doppelte Hochkommata verwendet, müssen das einfache Hochkommata sein, sonst umgekehrt.

  Beispiel:

  ```
  "COL=LENGTH,LENGTH_
  TAB='10=Abteilung,25=Leiter,10=Budget'"
  ```

- "DELIMITER"

  Die Spaltentrennung durch Zeichen wird in der Form **Delimiter** angegeben.
  "*" = frei wählbares Begrenzungszeichen. Die Begrenzungszeichen dienen nur zur Angabe des*Delimiter* und werden bei der Ausgabe nicht angezeigt.
  Delimiter = Zeichenkette aus maximal 10 Zeichen, welche die Spalten voneinander trennt. Dabei werden die Zeichen einer Zeile als Spalten zurückgegeben, die sich vor, zwischen oder nach der *Delimiter*-Zeichenkette befinden. Wenn *Delimiter* in einer Zeile nicht gefunden wird, erfolgt keine Spaltentrennung. Wird ein einzelnes Hochkomma als *Delimiter* verwendet, muss *Definition2* in doppelte Hochkommata eingeschlossen werden und umgekehrt.

  Standardwert: Semikolon (;)

UC_LOGIN	Name eines Login-Objektes. Format: Script-Literal oder Script-Variable ⚠ Beachten Sie, dass die Script-Funktion Anmeldedaten benötigt. Geben Sie den Parameter UC_LOGIN nicht an, so muss der entsprechende Ereignis-Job gültige Logininformationen enthalten.

Rückgabewert
Referenz auf die Datensequenz des Kommandos.

Anmerkungen

Die Script-Funktion PREP_PROCESS erzeugt eine Datensequenz, wie beispielsweise die Ausgaben von:

- Betriebssystemkommandos im BS2000, MPE, UNIX, VMS und Windows,
- BS2000-Consol-Kommandos,
- Abfragen des UNIX-Dateisystems,
- SAP-Monitoren, SAP-Systemlog und SAP-Jobs

ℹ Zum Ausführen eines BS2000-Consol-Kommandos und für Abfragen des UNIX-Dateisystems müssen Automation Engine-Hilfsprogramme (UCYEBXXZ bzw. UCXE???F) installiert sein.

Die Konsolen und Betriebssystem-Kommandos werden mit Hilfe von Ereignis-Jobs ausgeführt, die im Hintergrund auf dem Rechner des angegebenen Agenten (*Host*) gestartet werden. Die Ergebniszeilen eines Kommandos stehen als interne Liste (Datensequenz) über den Rückgabewert der Script-Funktion zur Verfügung.
Ereignis-Jobs besitzen den Namen "EVENT.*Ereignis-Job*". Dabei ist "EVENT." ein fest vorgegebener Bestandteil des Jobnamens, "*EreignisJob*" hingegen ist eine frei wählbare Bezeichnung. Die Definition eines solchen Ereignis-Jobs ist hinsichtlich bestimmter Attribute und des allgemeinen Scriptaufbaus vorgeschrieben.
Im Systemmandanten 0000 werden vorgefertigte Ereignis-Jobs ausgeliefert (Ordner: PREP_PROCESS). Sie können direkt oder als Vorlage im eigenen Mandanten verwendet werden. Der Inhalt der Ereignis-Jobs kann bei Bedarf angepasst werden. Beachten Sie, dass im EVENT.UNIXCMD der Scriptablauf zu ändern ist, wenn Sie nicht möchten, dass der Report beim Auftritt eines Rückgabewertes größer 0 gelöscht wird.

Bei der Bearbeitung der von PREP_PROCESS bereitgestellten Datensequenz werden folgende interne Arbeitsschritte durchlaufen:

1. Es wird der Ereignis-Job aktiviert, der mit dem Parameter *EreignisJob* angegeben wurde (Jobname: EVENT.*EreignisJob*).
2. Der Ereignis-Job läuft auf dem *Host* und führt eine im Parameter *Aktion* angegebene Aktion durch. Er leitet deren Ergebnis zeilenweise in eine Datensequenz um.
3. Es werden nur die Zeilen berücksichtigt, deren Inhalt dem Parameter *Filter* entspricht. Die Verwendung dieses Parameters ist optional.
4. Als Paramter *Aktion* können beliebige Wertzuweisungen angegeben werden.
   Bei den Betriebssystemkommandos im BS2000, MPE, UNIX, VMS und Windows sowie beim BS2000-Consol-Kommando wird mit *Variable* vorrangig die Script-Variable "&CMD" versorgt. Aber auch den Script-Variablen, die im Rahmen des Attribut-Dialogs zur Verfügung stehen, können Werte zugewiesen werden. v
   ⚠ Für die Abfrage des UNIX-Dateisystems gilt eine spezielle Syntax von *Aktion*.

5. Die auf dem *Host* erstellte Datensequenz wird zur Bearbeitung zur Automation Engine übertragen. Der Namen der Datei, welche vom *Host* zur Automation Engine übertragen wird, lautet standardmäßig "ERRRRRRR.TXT". Es werden folgende Platzhalter für die variablen Teile verwendet:

   - E - Ereignis
   - RRRRRRR - Laufende Nummer (RunID) der Aufgabe

   Beachten Sie, dass die RunID nicht als Zahl, sondern als umgewandelten Zeichenfolge dargestellt wird (siehe RUNNR2ALPHA).

Die Script-Funktion liest standardmäßig eine ganze Zeile. Es kann aber auch strukturiert zugegriffen werden, wenn die Zeile in Spalten unterteilt ist. Hierbei gelten folgende Bestimmungen:

- maximal 22 Spalten mit einer Gesamtlänge von 2048 Bytes,
- Spaltenbreite maximal 255 Zeichen und
- Spaltenname maximal 32 Zeichen.

Mit GET_PROCESS_LINE können Sie auf einzelne Spalten zugreifen.

Der Rückgabewert der Script-Funktion ist die Referenz auf die Datensequenz. Sie wird den Script-Anweisungen :PROCESS und :ENDPROCESS als Startparameter übergeben. In Kombination mit der Script-Funktion GET_PROCESS_LINE kann damit jede einzelne Zeile der Datensequenz und deren Spalten bearbeitet werden.

Enthält die Datensequenz den gesuchten Inhalt nicht, führt das zu keiner Fehlermeldung. Die Bearbeitung der Datensequenz, definiert zwischen :PROCESS und :ENDPROCESS, wird einfach nicht durchlaufen.

Der Script-Variablen, die die Datensequenz-Referenz enthält, kann kein neuer Wert zugewiesen werden. Erst wenn Sie die Datensequenz mit der Script-Anweisung :CLOSE_PROCESS explizit verwerfen, kann die Variable erneut verwendet werden.

Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

Beispiele

Im ersten Beispiel wird auf dem BS2000-Agenten "C70" ein Kommando "/STA P" abgesetzt. Die erforderliche Kennung und der Account werden aus dem Login-Objekt gelesen.

```
:SET  &HND# = PREP_PROCESS("C70","BS2000CMD",,"CMD=/STA P","UC_
LOGIN=ADMIN")
```

Dieses Beispiel setzt auf einer BS2000-Console ein Kommando zur Ausgabe aller eröffneten Anwendungen ab. Es wird kein Filter für den Zeileninhalt angegeben.

```
:SET  &HND# = PREP_PROCESS("C70","BS2000UCON",,"CMD=/BCDISP
DISP=O","UC_LOGIN=ADMIN")
```

Im dritten Beispiel werden die Verzeichnisse des Laufwerks C:\ auf dem Rechner des Windows-Agenten "WIN23" ermittelt.

```
:SET &HND# = PREP_PROCESS("WIN23","WINCMD","*DIR*","CMD=DIR C:","UC_
LOGIN=ADMIN")
```

Das vierte Beispiel liest den SAP-Monitor "MON1" aus dem Monitorset "AE" aus. Es soll auf die einzelnen Spalten der Monitordaten zugegriffen werden, die in der Datei definiert sind. Benutzer und SAP-Mandant werden aus dem angegebenen Login-Objekt gelesen.

```
:SET  &HND# = PREP_PROCESS
("T46","R3MONITOR","*","MONSET=AE","MONNAM=MON1","COL=FILE","UC_
LOGIN=AUTOMICADMIN")
```

Folgendes Beispiel führt einen Befehl auf dem UNIX-Agenten "UNIX01" aus, welcher Informationen über Verbindungen mit Port 2400 zurückliefert.

```
:SET &HND# = PREP_PROCESS("UNIX01","UNIXCMD",,"CMD=netstat -an | grep
2400","UC_LOGIN=LOGIN.UNIX")
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:CLOSE_PROCESS	Verwirft eine Datensequenz innerhalb eines Scripts.
:PROCESS... :TERM_ PROCESS... :ENDPROCESS	Definition einer Schleife zur zeilenweisen Bearbeitung einer Datensequenz, wie beispielsweise den Inhalt einer sequentiellen Datei oder das textmäßige Ergebnis eines Kommandos.
GET_PROCESS_ LINE	Ermittelt den aktuellen Zeileninhalt einer Datensequenz.

Script-Sprachmittel - Datensequenzen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.9.8 PREP_PROCESS_AGENTGROUP

Script-Funktion: Ermittelt mit Auswahlkriterien die Agenten eines AgentenGruppen-Objektes und stellt das Ergebnis als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.

Syntax

PREP_PROCESS_AGENTGROUP*(AgentenGruppe *[, Host] [, Selektierungsoption] [, RunID]**)***

Syntaxteil	Beschreibung/Format
AgentenGruppe	Name des AgentenGruppen-Objektes, dessen Agenten ausgelesen werden sollen. Format: AE-name, Script-Literal oder Script-Variable
Host	Filter für den Agentennamen Format: Script-Literal oder Script-Variable Maximal 32 Zeichen Standardwert: "*" Die Wildcardzeichen "*" und "?" können verwendet werden. "*" steht dabei für beliebige, "?" für genau ein Zeichen.

Selektierungsoption	Methode nach der die Agenten der AgentenGruppe ermittelt werden sollen. Format: AE-Name, Script-Literal oder Script-Variable Erlaubte Werte: "BY_RULE" (Standardwert), "ALL" und "RUNNR" "BY_RULE" - Die Script-Funktion liefert den Agenten zurück, auf dem die nächste Aufgabe laut dem Modus der AgentenGruppe laufen wird. "ALL" - Es werden alle Agenten der AgentenGruppe ermittelt. "RUNNR" - Der bzw. die ermittelten Agenten hängen von einer konkreten Aufgabe ab. Die RunID ihres AgentenGruppencontainers ist im gleichnamigen Parameter anzugeben. Bei AgentenGruppen mit dem Modus "Alle" liefern die Optionen "BY_RULE" und "ALL" dasselbe Ergebnis.
RunID	Laufende Nummer (RunID) des AgentenGruppencontainers. Format: Script-Literal, Script-Variable oder Zahl Bei der Verwendung dieses Parameters wird der bzw. die Agenten zurückgeliefert auf denen die Aufgabe gelaufen ist. Dieser Parameter ist nur dann anzugeben, wenn Sie die *Selektierungsoption* RUNNR ausgewählt haben.

Rückgabewert
Referenz auf die Datensequenz des AgentenGruppen-Objektes.

Anmerkungen

Die Script-Funktion liest Agenten eines AgentenGruppen-Objektes. Dabei können die ausgelesenen Agenten durch optionalen Parameter eingeschränkt werden. Standardmäßig wird die Referenz auf die gesamte Datensequenz geliefert.

Der Rückgabewert der Script-Funktion ist die Referenz auf eine Datensequenz. Sie wird den Script-Anweisungen :PROCESS und :ENDPROCESS als Startparameter übergeben. In Kombination mit der Script-Funktion GET_PROCESS_LINE kann auf jede einzelne Zeile der Datensequenz zugegriffen werden. Diese ist in zwei Spalten unterteilt die Sie auch gezielt auslesen können:

1. Name des Agenten
2. Status des Agenten ("Y" - Agent ist aktiv, "N" - Agent ist inaktiv)

Der Status ist vor allem bei AgentenGruppen relevant, die den Modus "Alle" eingestellt haben. Die Script-Funktion liefert für solche AgentenGruppen immer alle Agenten ohne darauf zu achten, ob sie aktiv sind oder nicht. Mit der Status-Information können Sie jene Agenten ermitteln auf denen Aufgaben tatsächlich durchgeführt werden können.

Wenn Sie nur nach der *Selektierung* und/oder der *RunID* filtern wollen, so sind die vorhergehenden Kommata auch zu setzen.

Der Script-Variablen, die die Datensequenz-Referenz enthält, kann kein neuer Wert zugewiesen werden. Erst wenn Sie die Datensequenz mit der Script-Anweisung :CLOSE_PROCESS explizit verwerfen, kann die Variable erneut verwendet werden.

Beispiel

Im ersten Beispiel werden alle Agenten einer AgentenGruppe ermittelt deren Name mit "WIN" beginnt.

```
:SET &HND# = PREP_PROCESS_AGENTGROUP("AGENTENGRUPPE_
WINDOWS","WIN*",ALL)

:PROCESS &HND#
:    SET &AGENT# = GET_PROCESS_LINE(&HND#,1)
:    SET  &STATUS# = GET_PROCESS_LINE(&HND#,2)
:    PRINT "Agent: &AGENT#"
:    PRINT "Status: &STATUS#"
:ENDPROCESS

:CLOSE_PROCESS &HND#
```

Im folgenden Beispiel wird ein Wiederanlauf einer Aufgabe durchgeführt. Die Script-Funktion SYS_ACT_ME_NR liefert in diesem Fall die RunID der Original-Durchführung mit der die RunID des AgentenGruppen-Containers ermittelbar ist. Dadurch können die Agenten auf denen die Aufgabe ursprünglich gelaufen sind, ausgelesen werden.

```
:SET  &A_RUNNR# = SYS_ACT_ME_NR()
:SET  &P_RUNNR# = GET_PARENT_NR(&A_RUNNR#)
:SET  &HND# = PREP_PROCESS_AGENTGROUP("AGENTENGRUPPE_DB",,RUNNR,&P_
RUNNR#)

:PROCESS &HND#
:    SET  &AGENT# = GET_PROCESS_LINE(&HND#,1)
:    SET &STATUS# = GET_PROCESS_LINE(&HND#,2)
:    PRINT "Agent: &AGENT#"
:    PRINT "Status: &STATUS#"
:ENDPROCESS

:CLOSE_PROCESS  &HND#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:CLOSE_PROCESS	Verwirft eine Datensequenz innerhalb eines Scripts.
:PROCESS... :TERM_ PROCESS... :ENDPROCESS	Definition einer Schleife zur zeilenweisen Bearbeitung einer Datensequenz, wie beispielsweise den Inhalt einer sequentiellen Datei oder das textmäßige Ergebnis eines Kommandos.
GET_PROCESS_ LINE	Ermittelt den aktuellen Zeileninhalt einer Datensequenz.

Script-Sprachmittel - Datensequenzen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.9.9 PREP_PROCESS_COMMENTS

Script-Funktion: Ermittelt mit Filtereinstellungen den Zeitstempel, Benutzer und Text von Aufgaben-Kommentaren und stellt das Ergebnis als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.

Syntax

PREP_PROCESS_COMMENTS*([RunID] [,Text] [, Benutzer])*

Syntaxteil	Beschreibung/Format
RunID	Laufende Nummer (RunID) der Aufgabe dessen Kommentare ausgelesen werden soll. Format: Script-Literal, Script-Variable oder Zahl Geben Sie keine *RunID* an, so werden die Kommentare jener Aufgabe ausgelesen, welche die Script-Anweisung aufruft.
Text	Filter für einen Ausdruck im Kommentartext Format: Script-Literal oder Script-Variable Maximal 200 Zeichen Standardwert: "*" Die Wildcardzeichen "*" und "?" können verwendet werden. "*" steht dabei für beliebige, "?" für genau ein Zeichen.
Benutzer	Filter für den Namen eines Benutzer-Objektes Format: Script-Literal oder Script-Variable Maximal 200 Zeichen Standardwert: "*" Die Wildcardzeichen "*" und "?" können verwendet werden. "*" steht dabei für beliebige, "?" für genau ein Zeichen. Wenn Sie nur nach dem Benutzer filtern wollen, ist das erste Komma auch zu setzen. Beispiel: `:SET &HND# = PREP_PROCESS_COMMENTS(,,"MEIER/UC4")`

Rückgabewert
Referenz auf die Datensequenz von Kommentaren

Anmerkungen

Die Script-Funktion liest die Kommentare einer Aufgabe aus. Dabei kann das Ergebnis mit den Parametern *Text* und *Benutzer* eingeschränkt werden.

⚠ Bei den Filterangaben wird die Groß- und Kleinschreibung berücksichtigt!

⚠ Beachten Sie, dass ein Leerstring "" als Filterangabe genauso wirkt wie "*", d.h. alle Werte werden zurückgeliefert.

Der Rückgabewert der Script-Funktion ist die Referenz auf eine Datensequenz. Sie wird den Script-Anweisungen :PROCESS und :ENDPROCESS als Startparameter übergeben. In Kombination mit

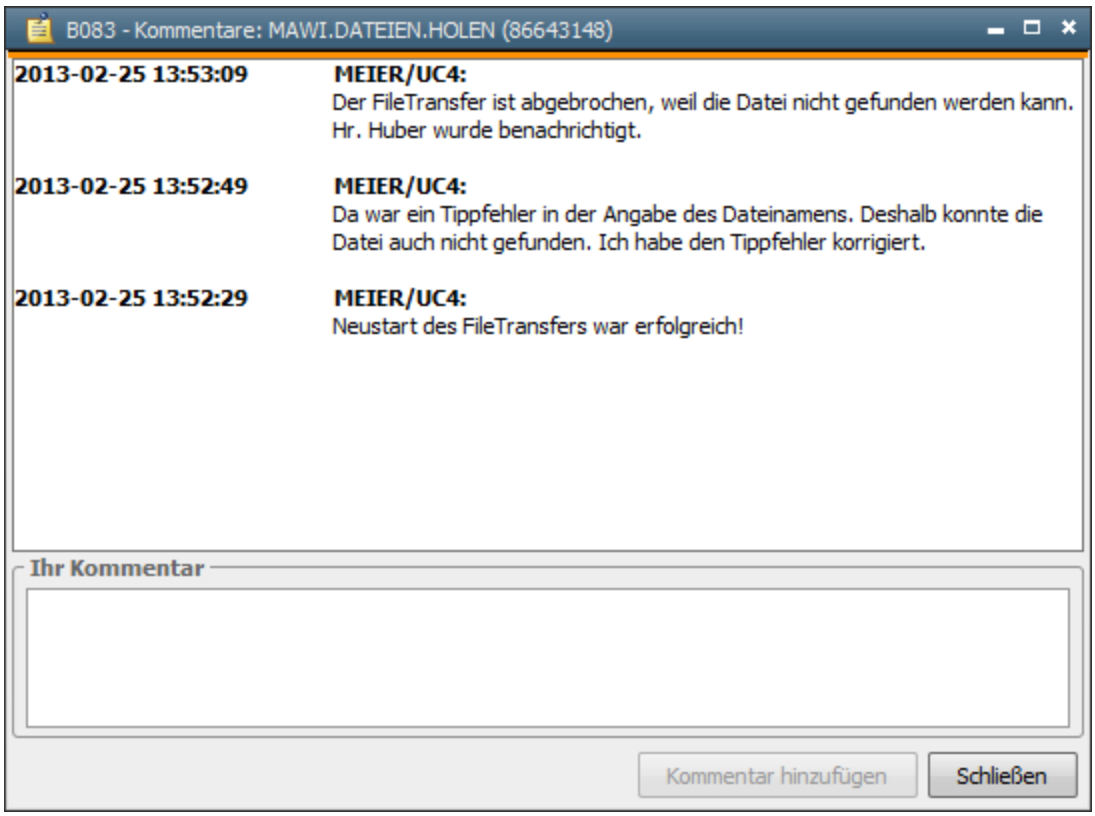
der Script-Funktion GET_PROCESS_LINE kann nun auf jeden Kommentareintrag der Aufgabe zugegriffen werden.

Enthält der Kommentar die gesuchten Werte nicht, führt das zu keiner Fehlermeldung. Die Bearbeitung der Datensequenz, definiert zwischen :PROCESS und :ENDPROCESS, wird einfach nicht durchlaufen.

Der Script-Variablen, die die Datensequenz-Referenz enthält, kann kein neuer Wert zugewiesen werden. Erst wenn Sie die Datensequenz mit der Script-Anweisung :CLOSE_PROCESS explizit verwerfen, kann die Variable erneut verwendet werden.

Beispiele

Gegeben sind die folgenden Kommentare:


B083 - Kommentare: MAWI.DATEIEN.HOLEN (86643148)
2013-02-25 13:53:09
MEIER/UC4:
Der FileTransfer ist abgebrochen, weil die Datei nicht gefunden werden kann.
Hr. Huber wurde benachrichtigt.
2013-02-25 13:52:49
MEIER/UC4:
Da war ein Tippfehler in der Angabe des Dateinamens. Deshalb konnte die
Datei auch nicht gefunden. Ich habe den Tippfehler korrigiert.
2013-02-25 13:52:29
MEIER/UC4:
Neustart des FileTransfers war erfolgreich!
Ihr Kommentar
Kommentar hinzufügen
Schließen

Das erste Beispiel ermittelt alle Kommentare und gibt den Benutzer sowie den Text mit :PRINT aus.

```
:SET  &HND# = PREP_PROCESS_COMMENTS()

:PROCESS  &HND#
:   SET  &BENUTZER# = GET_PROCESS_LINE(&HND#,2)
:   SET  &TEXT# = GET_PROCESS_LINE(&HND#,3)
:   PRINT  "&BENUTZER#:  &TEXT#"
:ENDPROCESS

:CLOSE_PROCESS &HND#
```

Im zweiten Beispiel werden nur die Einträge von Hrn. Meier gelesen und samt Zeitstempel ausgegeben.

```
:SET  &HND# = PREP_PROCESS_COMMENTS(,,"MEIER/UC4")
```

```
:PROCESS  &HND#
:    SET  &ZEIT# = GET_PROCESS_LINE(&HND#,1)
:    SET  &TEXT# = GET_PROCESS_LINE(&HND#,3)
:    PRINT  "&ZEIT#:  &TEXT#"
:ENDPROCESS

:CLOSE_PROCESS &HND#
```

Im dritten Beispiel wird die Script-Funktion aus einer anderen Aufgabe aufgerufen. Die RunID wurde daher angegeben. Als Ergebnis werden alle Kommentartexte geliefert, die das Wort "Datei" enthalten.

```
:SET  &RUNID# = GET_UC_OBJECT_NR(MAWI.DATEIEN.HOLEN)
:SET  &HND# = PREP_PROCESS_COMMENTS(&RUNID#,"*Datei*")

:PROCESS  &HND#
:    SET  &TEXT# = GET_PROCESS_LINE(&HND#,3)
:    PRINT  "Kommentar: &TEXT#"
:ENDPROCESS

:CLOSE_PROCESS &HND#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:ADD_COMMENT	Fügt einer Aufgabe einen Kommentar hinzu.
:CLOSE_PROCESS	Verwirft eine Datensequenz innerhalb eines Scripts.
:PROCESS... :TERM_ PROCESS... :ENDPROCESS	Definition einer Schleife zur zeilenweisen Bearbeitung einer Datensequenz, wie beispielsweise den Inhalt einer sequentiellen Datei oder das textmäßige Ergebnis eines Kommandos.
GET_PROCESS_ LINE	Ermittelt den aktuellen Zeileninhalt einer Datensequenz.

Script-Sprachmittel - Datensequenzen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.9.10 PREP_PROCESS_DOCU

Script-Funktion: Stellt den Inhalt einer Dokumentations-Registerkarte als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.

Syntax

PREP_PROCESS_DOCU(***Objektname*** *[, Doku-Tab]*)

Syntaxteil	Beschreibung/Format

Objektname	Name eines beliebigen Objektes, dessen Dokumentations-Registerkarte ausgelesen werden soll. Format: Script-Literal oder Script-Variable
Doku-Tab	Name der Dokumentations-Registerkarte Format: Script-Literal oder Script-Variable Erfolgt keine Angabe, so wird nach einer Dokumentations-Registerkarte mit dem Namen "Docu" gesucht.

Rückgabewerte
Referenz auf die Datensequenz, welche den Inhalt der gewünschten Dokumentation-Registerkarte enthält.

Anmerkungen

Das Sprachmittel liest den Inhalt einer bestimmten Dokumentations-Registerkarte eines Objektes aus (Dokumentations-Registerkarten stehen für alle Objekttypen zur Verfügung) und stellt diesen in Form einer Datensequenz für die weitere Verarbeitung bereit. Anzugeben ist der Name des Objektes und der Dokumentations-Registerkarte (wie in der Variable UC_OBJECT_DOCU definiert).

Dabei kann sowohl eine strukturierte als auch eine einfache Doku-Registerkarte angegeben werden (= enthält nur Text).
Bei einer strukturierten Doku-Registerkarte, werden alle Elemente, die im linken Bereich der Registerkarte definiert sind, zeilenweise in die Datensequenz abgespeichert. Wird eine einfache Doku-Registerkarte angegeben, so wird deren Inhalt (Text) zeilenweise in die Datensequenz umgeleitet.

Bei der Angabe des Namens der Dokumentations-Registerkarte spielt Groß- und Kleinschreibung keine Rolle. Beispiel: Wird nach "docu" gesucht, wird die Registerkarte "Docu" trotzdem gefunden.

Wird keine Dokumentations-Registerkarte mit dem angegebenen Namen gefunden, bricht die Script-Funktion nicht ab. Die Datensequenz ist in diesem Fall lediglich nicht befüllt. Zum Abbruch kommt es nur, wenn der Objektname ungültig ist.

Beispiel

In folgendem Beispiel wird die Dokumentations-Registerkarte "Docu" des Objektes SCRI.TEST ausgelesen und deren Inhalt im Aktivierungsreport ausgegeben.

```
:SET &HND# = PREP_PROCESS_DOCU(SCRI.TEST,Docu)
:PROCESS &HND#
: SET &LINE# = GET_PROCESS_LINE(&HND#)
: PRINT &LINE#
:ENDPROCESS

:CLOSE_PROCESS &HND#
```

Siehe auch:

Script-Sprachmittel	Beschreibung

:PROCESS... :TERM_ PROCESS... :ENDPROCESS	Definition einer Schleife zur zeilenweisen Bearbeitung einer Datensequenz, wie beispielsweise den Inhalt einer sequentiellen Datei oder das textmäßige Ergebnis eines Kommandos.
GET_PROCESS_ LINE	Ermittelt den aktuellen Zeileninhalt einer Datensequenz.

Script-Sprachmittel - Datensequenzen
Beispielsammlung
Endestatus aufgrund des Reportinhaltes setzen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.9.11 PREP_PROCESS_FILE

Script-Funktion: Ermittelt mit Filterkriterien zeilenweise den Inhalt einer Textdatei, die sich auf einem angegebenen Rechner befindet, und stellt das Ergebnis als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.

Syntax

PREP_PROCESS_FILE*(Host, Datei,* *[Filter][, "***COL=***Definition1[, Definition2]"][,* **UC_ LOGIN=***Login-Objekt])*

Syntaxteil	Beschreibung/Format
Host	Rechner (Name des Agenten), auf dem sich die Datei befindet. Format: Script-Literal oder Script-Variable
Datei	Name der Textdatei mit kompletter Pfadangabe, die gelesen werden soll. Format: Script-Literal oder Script-Variable Der Name einer Datei, die zu einer Generation Data Group gehört, können Sie mit der Script-Funktion GET_FILESYSTEM ermitteln.
Filter	Vorgabe für den Zeileninhalt. Groß- und Kleinschreibung wird nicht beachtet. Format: Script-Literal oder Script-Variable Standardwert: "*" Die Wildcardzeichen "*" und "?" können verwendet werden. "*" steht dabei für beliebige, "?" für genau ein Zeichen.

Definition1	Legt fest, ob die Zeilen in Spalten unterteilt sein sollen. Format: Script-Literal oder Script-Variable Erlaubte Werte: "NONE" (Standardwert), "FILE", "LENGTH", "DELIMITER" "NONE" = Keine Unterteilung in Spalten. "FILE" = Die Spalten wurden in der Datei definiert. "LENGTH" = Vorgegebene Spaltenbreiten. Erfordert LENGTH_TAB= als *Definition2*. "DELIMITER" = Spalten werden durch ein Begrenzungszeichen getrennt. Erfordert DELIMITER= als *Definition2*.

Definition2	Legt Spaltenbreiten und Spaltennamen (optional) oder das Begrenzungszeichen fest.
	Format: Script-Literal oder Script-Variable
	
	Erlaubte Werte: "LENGTH_TAB" und "DELIMITER"
	
	• "LENGTH_TAB"
	
	Spaltenbreiten und Spaltennamen (optional) werden in der Form *Spaltenbreite*=[*Spaltenname*] angegeben. Die Spaltenbreite wird über eine Zeichenanzahl definiert. Die einzelnen Spalten sind durch Kommata zu trennen (maximal 22 Spalten). Vor der ersten und nach der letzten Spaltendefinition müssen zusätzlich Hochkommata stehen. Werden für *Definition2* als Script-Literal doppelte Hochkommata verwendet, müssen das einfache Hochkommata sein, sonst umgekehrt.
	
	Beispiel:
	
	`"COL=LENGTH,LENGTH_TAB='3=drive,100=file name'"`
	
	• "DELIMITER"
	
	Das Begrenzungszeichen wird in der Form **Delimiter** angegeben.
	"*" = frei wählbares Begrenzungszeichen.
	Delimiter = Zeichenkette aus maximal 10 Zeichen, welche die Spalten voneinander trennt. Wird ein einzelnes Hochkomma als *Delimiter* verwendet, muss der gesamte Ausdruck **COL=***Definition1[, Definition2]* in doppelte Hochkommata eingeschlossen werden und umgekehrt.
	
	Standardwert: Semikolon (;)
	
	Beispiel:
	
	`"COL=DELIMITER,DELIMITER=*'*"`
	`'COL=DELIMITER,DELIMITER=@"@'`
	
	Als Trennzeichen können Sie auch Tabulatoren definieren:
	
	`'DELIMITER=<TAB>'`

UC_LOGIN	Name eines Login-Objektes. Format: Script-Literal oder Script-Variable

Rückgabewert
Referenz auf die Datensequenz der Datei.

Anmerkungen

Die Script-Funktion PREP_PROCESS_FILE stellt den Inhalt einer Textdatei, beispielsweise LOG- oder Trace-Datei, für eine weitere Verarbeitung mit AE Script bereit.

Die Script-Funktion liest standardmäßig eine ganze Zeile. Es kann aber auch strukturiert zugegriffen werden, wenn die Zeile in Spalten unterteilt ist. Hierbei gelten folgende Bestimmungen:

- maximal 22 Spalten mit einer Gesamtlänge von 2048 Bytes,
- Spaltenbreite maximal 255 Zeichen und
- Spaltenname maximal 32 Zeichen.

Die Spalten können auch in der Datei selbst definiert werden (erste Zeile). Beispiel:

```
COL=LENGTH,LENGTH_TAB='74=PATH,25=NAME,5=VALUE,2=STATUS,9=DATE,7=TIME'
```

Mit GET_PROCESS_LINE können Sie auf einzelne Spalten zugreifen.

Mit dem optionalen Parameter UC_LOGIN kann der Script-Funktion der Name eines Login-Objektes übergeben werden. Der Zugriff auf die zu lesende Datei und deren Übertragung vom Host zur Automation Engine wird mit den im Login definierten Anmeldedaten durchgeführt. Benutzer benötigen das Privileg "FileTransfer: Ohne Angabe von Benutzerkennung ausführen", wenn die Script-Funktion PREP_PROCESS_FILE ohne den Parameter UC_LOGIN verwendet werden soll.

Der Rückgabewert der Script-Funktion ist die Referenz auf eine Datensequenz. Sie wird den Script-Anweisungen :PROCESS und :ENDPROCESS als Startparameter übergeben. In Kombination mit der Script-Funktion GET_PROCESS_LINE kann damit jede einzelne Zeile der Datensequenz und deren Spalten bearbeitet werden.

Enthält die Datei den gesuchten Inhalt nicht, führt das zu keiner Fehlermeldung. Die Bearbeitung der Datensequenz, definiert zwischen :PROCESS und :ENDPROCESS, wird einfach nicht durchlaufen.

Der Script-Variablen, die die Datensequenz-Referenz enthält, kann kein neuer Wert zugewiesen werden. Erst wenn Sie die Datensequenz mit der Script-Anweisung :CLOSE_PROCESS explizit verwerfen, kann die Variable erneut verwendet werden.

PREP_PROCESS_FILE wird nur von Betriebssystem-Agenten unterstützt. Die Verwendung im Zusammenhang mit Agenten für Datenbanken, Oracle Applications, PeopleSoft, SAP, JMX oder Rapid Automation ist nicht möglich.

Beachten Sie, dass die Verarbeitung von äußerst großen Dateien Einfluss auf die Performance des Agenten hat.

Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

Beispiele

Das erste Beispiel gibt alle Zeilen, die den Begriff "Start" enthalten, im Aktivierungsprotokoll aus.

```
:SET &HND#=PREP_PROCESS_FILE
(WIN21,"\\FServer\UC4\BSP\INPUT.TXT","*Start*")
:PROCESS &HND#
:    SET &LINE#=GET_PROCESS_LINE(&HND#)
:    PRINT &LINE#
:ENDPROCESS
```

Das zweite Beispiel liest alle Zeilen einer Datei aus, während der Parameter "COL-FILE" angibt, dass die Spaltenbezeichnungen und Spaltenbreite in der Datei selbst definiert sind. Das entsprechende Login-Objekt wird zum Login genutzt.

```
:SET &HND# = PREP_PROCESS_FILE(WIN21, "\\FServer\LOG.TXT",
 ,"COL=FILE",'UC_LOGIN=UC4FT')
```

Das dritte Beispiel ähnelt dem vorigen, allerdings gibt der Parameter "COL-LENGTH" an, dass die Spaltendefinitionen mit dem nachfolgenden Parameter angegeben werden.
Spaltendefinitionen können einer Spaltenbezeichnung zugeordnete Spaltenbreiten sein, oder nur die Spaltenbreite.
Spalten ohne Bezeichnung werden ignoriert.

```
:SET &HND# = PREP_PROCESS_FILE(WIN21, "\\FServer\UC4\DIALOG\TEMP\UCDJ_
LOGG_01.TXT","*DB-INFO*","COL=LENGTH,LENGTH_
TAB='8=DATUM,1,6=ZEIT,7,200=TEXT'")
```

Im nächsten Beispiel zeigt der Parameter "COL-DELIMITER" an, dass ein Begrenzungszeichen mithilfe des folgenden Parameters definiert wird – der Tabulator. Das Script gibt nur die dritte Spalte aus.

```
:SET &HND# = PREP_PROCESS_FILE(UNIX01,
"/uc4/test.txt",,"COL=DELIMITER,DELIMITER=*<TAB>*")
:PROCESS &HND#
:    SET &LINE# = GET_PROCESS_LINE(&HND#,3)
:    PRINT &LINE#
:ENDPROCESS
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:CLOSE_PROCESS	Verwirft eine Datensequenz innerhalb eines Scripts.
:PROCESS... :TERM_ PROCESS... :ENDPROCESS	Definition einer Schleife zur zeilenweisen Bearbeitung einer Datensequenz, wie beispielsweise den Inhalt einer sequentiellen Datei oder das textmäßige Ergebnis eines Kommandos.
GET_PROCESS_ LINE	Ermittelt den aktuellen Zeileninhalt einer Datensequenz.

Script-Sprachmittel - Datensequenzen
Allgemeines zum Script
Script-Sprachmittel - funktionelle Gliederung
Script-Sprachmittel - alphabetische Auflistung

3.9.12 PREP_PROCESS_FILENAME

Script-Funktion: Ermittelt eine Liste mit Namen von Dateien, die sich in dem angegebenen Verzeichnis eines Rechners befinden, und stellt diese als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.

Syntax

PREP_PROCESS_FILENAME*(Host, Dateiname,* *[Wildcards], [Unterordner], [Filter] [,"***COL=***Definition1[, Definition2]"] [,"***UC_LOGIN=***Login-Objekt"]* ***)***

Syntaxteil	Beschreibung/Format
Host	Rechner (Name des Agenten), auf dem sich die Dateien befinden. Format: Script-Literal oder Script-Variable
Dateiname	Pfad und Dateiname, nach dem gesucht werden soll. Format: Script-Literal oder Script-Variable Im Dateinamen können die Wildcardzeichen "*" und "?" verwendet werden. "*" steht dabei für beliebige, "?" für genau ein Zeichen. In Pfaden sind keine Wildcardzeichen erlaubt. Ob die Groß- und Kleinschreibung des Dateinamens berücksichtigt wird, hängt vom Parameter *Wildcards* ab.
Wildcards	Kennzeichen, ob der Dateiname mit Wildcard-Zeichen angegeben wurde. Format: Script-Literal oder Script-Variable der Parameter auch Auswirkungen auf die Prüfung der Schreibweise des Datenamens hat! Erlaubte Werte: "Y" (Standardwert) und "N" "Y" = Dateiname mit Wildcardzeichen, Schreibweise wird nicht berücksichtigt "N" = Dateiname ohne Wildcardzeichen, Schreibweise wird berücksichtigt
Unterordner	Kennzeichen, ob Unterordner bzw. -verzeichnisse in die Suche einbezogen werden sollen. Format: Script-Literal oder Script-Variable Erlaubte Werte: "Y" und "N" (Standardwert) "Y" = Unterordner bzw. -verzeichnisse einbeziehen. "N" = Unterordner bzw. -verzeichnisse nicht berücksichtigen.
Filter	Zusätzliche Möglichkeit, die Zeilen der Datensequenz zu filtern. Format: Script-Literal oder Script-Variable Groß- und Kleinschreibung wird nicht beachtet. Die Wildcardzeichen "*" und "?" können verwendet werden. "*" steht dabei für beliebige, "?" für genau ein Zeichen. Standard: "*"

Definition1	Legt fest, ob die Zeilen der Datensequenz in Spalten unterteilt sind. Format: Script-Literal oder Script-Variable Erlaubte Werte: "NONE" (Standardwert), "LENGTH", "DELIMITER" "NONE" = Keine Unterteilung in Spalten. "LENGTH" = Vorgegebene Spaltenbreiten. Erfordert LENGTH_TAB= als *Definition2*. "DELIMITER" = Spalten werden durch ein Begrenzungszeichen getrennt. Erfordert DELIMITER= als *Definition2*.

Definition2	Legt Spaltenbreiten und Spaltennamen (optional) oder das Begrenzungszeichen fest.
	Format: Script-Literal oder Script-Variable
	
	Erlaubte Werte: "LENGTH_TAB" und "DELIMITER"
	
	• "LENGTH_TAB"
	
	Spaltenbreiten und Spaltennamen (optional) werden in der Form *Spaltenbreite*=[*Spaltenname*] angegeben. Die Spaltenbreite wird über eine Zeichenanzahl definiert. Die einzelnen Spalten sind durch Kommata zu trennen (maximal 22 Spalten). Vor der ersten und nach der letzten Spaltendefinition müssen zusätzlich Hochkommata stehen. Werden für *Definition2* als Script-Literal doppelte Hochkommata verwendet, müssen das einfache Hochkommata sein, sonst umgekehrt.
	
	Beispiel:
	
	`"COL=LENGTH,LENGTH_TAB='3=drive,100=file name'"`
	
	• "DELIMITER"
	
	Das Begrenzungszeichen wird in der Form **Delimiter** angegeben.
	"*" = frei wählbares Begrenzungszeichen.
	Delimiter = Zeichenkette aus maximal 10 Zeichen, welche die Spalten voneinander trennt. Wird ein einzelnes Hochkomma als *Delimiter* verwendet, muss der gesamte Ausdruck **COL=***Definition1[, Definition2]* in doppelte Hochkommata eingeschlossen werden und umgekehrt.
	
	Standardwert: Semikolon (;)
	
	Beispiel:
	
	`"COL=DELIMITER,DELIMITER=*'*"`
	`'COL=DELIMITER,DELIMITER=@"@'`

Login-Objekt	Name eines Login-Objektes. Format: Script-Literal oder Script-Variable Der gesamte Ausdruck **UC_LOGIN=***Login-Objekt* muss unter Hochkomma gesetzt werden.

Rückgabewert
Referenz auf die Datensequenz der Dateiliste. "20240" - Fehlende Login-Information "20303" - AE-Berechtigungsfehler "20349" - Agent nicht verfügbar "20510" - Es wurde ein CP/WP anstelle eines Agenten angegeben. "20514" - Ungültiger Angabe bei Verwendung von Wildcards

Anmerkungen

Die Script-Funktion PREP_PROCESS_FILENAME stellt eine Liste von Dateinamen für eine weitere Verarbeitung mit AE Script bereit. Die Dateien werden auf dem Host durch den Agenten ermittelt und als Datensequenz aufbereitet. Die Auflistung erfolgt in alphabetisch aufsteigender Reihenfolge der Dateinamen.

Welche Dateien aufgelistet werden sollen, wird der Script-Funktion mit dem Parameter *Dateiname* mitgegeben. Enthält *Dateiname* eine Laufwerksbezeichnung, werden die gefundenen Dateien auch mit Laufwerksbezeichnung aufgelistet. Es kann festgelegt werden, ob Unterordner bzw. -verzeichnisse berücksichtigt werden sollen. Die Liste der gefundenen Dateien kann zusätzlich gefiltert werden.

Wildcards, *Unterordner*, *Filter* und die Parameter zur Unterteilung in Spalten sind optional. Wird ein Parameter in dieser Reihenfolge ausgelassen, muss trotzdem dessen Komma gesetzt werden. Der Aufruf der Script-Funktion kann mit jedem optionalen Parameter enden. Danach sind keine Kommata notwendig. *Definition2* darf nicht ohne *Definition1* verwendet werden.

Die Script-Funktion GET_PROCESS_LINE liest standardmäßig die gesamte Zeile einer Datensequenz. Es kann aber auch strukturiert zugegriffen werden, wenn die Zeile in Spalten unterteilt ist. Hierbei gelten folgenden Bestimmungen:

- maximal 22 Spalten mit einer Gesamtlänge von 2048 Bytes,
- Spaltenbreite maximal 255 Zeichen und
- Spaltenname maximal 32 Zeichen.

Mit GET_PROCESS_LINE können Sie auf einzelne Spalten zugreifen.

Mit dem optionalen Parameter UC_LOGIN kann der Script-Funktion der Name eines Login-Objektes übergeben werden. PREP_PROCESS_FILENAME verwendet die im Login definierten Anmeldedaten. Dadurch ist beispielsweise ein Zugriff auf verbundene Netzlaufwerke möglich. Benutzer benötigen das Privileg "FileTransfer: Ohne Angabe von Benutzerkennung ausführen", wenn die Script-Funktion PREP_PROCESS_FILENAME ohne den Parameter UC_LOGIN verwendet werden soll.

Der Rückgabewert der Script-Funktion ist die Referenz auf eine Datensequenz. Sie wird den Script-Anweisungen :PROCESS und :ENDPROCESS als Startparameter übergeben. In Kombination mit der Script-Funktion GET_PROCESS_LINE kann damit jede einzelne Zeile der Datensequenz und deren Spalten bearbeitet werden.

Enthält die Dateiliste den gesuchten Inhalt nicht, führt das zu keiner Fehlermeldung. Die Bearbeitung der Datensequenz, definiert zwischen :PROCESS und :ENDPROCESS, wird einfach nicht durchlaufen.

Der Script-Variablen, die die Datensequenz-Referenz enthält, kann kein neuer Wert zugewiesen werden. Erst wenn Sie die Datensequenz mit der Script-Anweisung :CLOSE_PROCESS explizit verwerfen, kann die Variable erneut verwendet werden.

⚠ PREP_PROCESS_FILE wird nicht von den Agenten für Anwendungen (Oracle Applications, PeopleSoft oder SAP) unterstützt.

Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

Mit der Script-Anweisung :ON_ERROR legen Sie die Reaktion auf Fehler fest, die bei der Script-Funktion auftreten. Sie können den Fehler mit den Script-Funktionen zur Fehlerbehandlung analysieren. Das Script wird standardmäßig weiter prozessiert. Es besteht aber auch die Möglichkeit, die Prozessierung des Scripts abzubrechen.

Beispiele

Das Beispiel listet alle HTML-Dokumente aus der WebHelp der AE-Dokumentation auf. Es wird explizit angegeben, dass Wildcardzeichen verwendet werden.

```
:SET &HND# = PREP_PROCESS_FILENAME
("WIN01","c:\AUTOMIC\documentation\webhelp\german\uc*.htm","Y",,,,"UC_
LOGIN=WIN_LOGIN")
:PROCESS &HND#
:    SET &LINE#=GET_PROCESS_LINE(&HND#)
:    PRINT &LINE#
:ENDPROCESS
```

Das zweite Beispiel erzeugt eine Liste aller Stylesheets, welche mit der AE-Dokumentation ausgeliefert werden. Alle Unterordner des Dokumentationsverzeichnisses sollen durchsucht werden. Das Ergebnis enthält keine Laufwerksangabe und wird im Report ausgegeben.

```
:SET &HND# = PREP_PROCESS_FILENAME("WIN01",
"\AUTOMIC\documentation\uc*.css","Y","Y",)
:PROCESS &HND#
:    SET &LINE#=GET_PROCESS_LINE(&HND#)
:    PRINT &LINE#
:ENDPROCESS
```

Im dritten Beispiel wird eine Liste mit allen AE-Programmen erstellt. Die einzelnen Zeilen sind in Spalten aufgeteilt. Die Spalten werden durch den Backslash getrennt. Bei der Verarbeitung der Datensequenz wird auf die 5. Spalte zugegriffen, welche die Dateinamen der Programme enthält.

```
:SET &HND# = PREP_PROCESS_FILENAME
(
"WIN01"
,"c:\AUTOMIC\server\bin\*.exe",,,,"COL=DELIMITER,DELIMITER=*\*")
:PROCESS &HND#
: SET &LINE#=GET_PROCESS_LINE(&HND#,5)
: PRINT &LINE#
:ENDPROCESS
```

Das vierte Beispiel basiert auf dem dritten. Hier wird aber ein Filter gesetzt, so dass nur der Dateiname der Automation Engine im Aktivierungsprotokoll ausgegeben wird.

```
:SET &HND# = PREP_PROCESS_FILENAME
(
"WIN01"
,
"c:\AUTOMIC\server\bin\*.exe"
,,,"*server*","COL=DELIMITER,DELIMITER=*\*")
:PROCESS &HND#
: SET &LINE#=GET_PROCESS_LINE(&HND#,5)
: PRINT &LINE#
:ENDPROCESS
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:CLOSE_PROCESS	Verwirft eine Datensequenz innerhalb eines Scripts.
:PROCESS... :TERM_ PROCESS... :ENDPROCESS	Definition einer Schleife zur zeilenweisen Bearbeitung einer Datensequenz, wie beispielsweise den Inhalt einer sequentiellen Datei oder das textmäßige Ergebnis eines Kommandos.
GET_PROCESS_ LINE	Ermittelt den aktuellen Zeileninhalt einer Datensequenz.

Script-Sprachmittel - Datensequenzen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.9.13 PREP_PROCESS_PROMPTSET

Script-Funktion: Liest die Definition von PromptSet-Objekten aus und stellt diese in Form einer internen Auflistung (Datensequenz) für die weitere Verarbeitung bereit.

Syntax

PREP_PROCESS_PROMPTSET*(Objektname)*

Syntaxteil	Beschreibung/Format
Objektname	Name des PromptSet-Objektes, deren Definition ermittelt werden soll. Format: Script-Literal oder Script-Variable

Rückgabewerte
Referenz auf die Datensequenz, welche die PromptSet-Definition enthält.

Anmerkungen

Die Script-Funktion dient zum Auslesen der Definition von PromptSet-Objekten und stellt diese in Form einer Datensequenz zur Verfügung. Anzugeben ist der Name des PromptSet-Objektes, zurückgeliefert wird die Referenz auf die entsprechende Datensequenz.

Für jedes Element des PromptSet-Objektes enthält die Datensequenz eine Zeile. Die Spalten jeder Zeile speichern die unterschiedlichen Informationen. Der Zugriff auf diese Spalten ist mit der Script-Funktion GET_PROCESS_LINE möglich.

Konkret werden pro Element folgende Informationen ermittelt:

- Name der PromptSet-Variable (ohne führendes &)
- Wert des benutzerdefinierten Feldes
- Wert des PromptSet-Elementes (Standardwert)

- Typ des PromptSet-Elementes (text, integer usw.)
- Objektname des Referenz-Objektes (Variable oder Kalender)
- Begriff des Kalender-Objektes (falls vorhanden)

Die Reihenfolge dieser Auflistung entspricht der Spaltenanordnung.

Wird bei GET_PROCESS_LINE keine bestimmte Spalte angegeben, so alle Informationen des Elementes getrennt durch die Zeichen "§§§" zurückgegeben.

Beispiel

Folgendes Beispiel-Script liest die Definition des PromptSet-Objektes PRPT.JOBP und gibt die Zeilen im Aktivierungsreport aus. Dabei werden alle Information der Elemente ermittelt.

```
:SET &HND# = PREP_PROCESS_PROMPTSET(PRPT.JOBP)
:PROCESS &HND#
: SET &LINE# = GET_PROCESS_LINE(&HND#)
: P &LINE#
:ENDPROCESS

:CLOSE_PROCESS &HND#
```

In das Aktivierungsprotokoll werden dadurch zum Beispiel folgende Informationen ausgegeben:

```
2013-01-31 11:28:59 - U0020408 TEXTFIELD1§§§§§§§§§text§§§VARA.JOB§§§
2013-01-31 11:28:59 - U0020408 INTEGER1§§§§§§0§§§integer§§§UC_
DATATYPE_NUMERIC§§§
2013-01-31 11:28:59 - U0020408 COMBOBOX1§§§§§§§§§combo§§§DB_WARTUNG§§§
2013-01-31 11:28:59 - U0020408 RADIOGROUP1§§§§§§§§§dynradiogroup§§§UC_
OBJECT_TEMPLATE§§§
2013-01-31 11:28:59 - U0020408 CHECKGROUP1§§§§§§§§§dyncheckgroup§§§UC_
SENDTO_ACT§§§
2013-01-31 11:28:59 - U0020408 CHECKLIST2§§§§§§§§§dyncheckgroup§§§UC_
UTILITY_ARCHIVE§§§
2013-01-31 11:28:59 - U0020408 DATE1§§§§§§2010-09-03§§§datefield§§§UC_
DATATYPE_DATE§§§0
2013-01-31 11:28:59 - U0020408 TIMESTAMP2§§§§§§2010-10-28
14:34:57§§§timestamp§§§UC_DATATYPE_TIMESTAMP§§§0
```

Das zweite Beispiel liest nur den Referenz-Objektenamen und den Typ der PromptSet-Elemente aus.

```
:SET &HND# = PREP_PROCESS_PROMPTSET(PRPT.JOBP)
:PROCESS &HND#
: SET &VAR# = GET_PROCESS_LINE(&HND#,1)
: SET &TYP# = GET_PROCESS_LINE(&HND#,4)
: P &VAR# &TYP#
:ENDPROCESS

:CLOSE_PROCESS &HND#
```

Dies Ausgabe würde dadurch beispielsweise so aussehen:

```
2013-01-31 11:59:20 - U0020408 TEXTFIELD1 text
2013-01-31 11:59:20 - U0020408 INTEGER1 integer
2013-01-31 11:59:20 - U0020408 COMBOBOX1 combo
2013-01-31 11:59:20 - U0020408 RADIOGROUP1 dynradiogroup
2013-01-31 11:59:20 - U0020408 CHECKGROUP1 dyncheckgroup
```

```
2013-01-31 11:59:20 - U0020408 CHECKLIST2 dyncheckgroup
2013-01-31 11:59:20 - U0020408 DATE1 datefield
2013-01-31 11:59:20 - U0020408 TIMESTAMP2 timestamp
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:PROCESS... :TERM_ PROCESS... :ENDPROCESS	Definition einer Schleife zur zeilenweisen Bearbeitung einer Datensequenz, wie beispielsweise den Inhalt einer sequentiellen Datei oder das textmäßige Ergebnis eines Kommandos.
GET_PROCESS_ LINE	Ermittelt den aktuellen Zeileninhalt einer Datensequenz.

Script-Sprachmittel - Datensequenzen
Beispielsammlung
Endestatus aufgrund des Reportinhaltes setzen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.9.14 PREP_PROCESS_REPORT

Script-Funktion: Ermittelt mit Filterkriterien Reportzeilen von aktivierbaren Objekten und stellt das Ergebnis als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.

Syntax

PREP_PROCESS_REPORT(*[Objekttyp],[RunID],[Reporttyp] [,Filter] [,"***COL***=Definition1] [,Definition2]"*)

Syntaxteil	Beschreibung/Format
Objekttyp	Kurzbezeichnung des Typs des Objekts, das der Klasse der aktivierbaren Objekte angehört. Es kann auch der Report des Mandanten, in dem diese Script-Funktion verwendet wird, aufbereitet werden. Die Kurzbezeichnung lautet in diesem Fall "CLNT". Format: AE-name, Script-Literal oder Script-Variable Optionaler Parameter, da der Objekttyp durch die laufende Nummer (RunID) eindeutig zugeordnet werden kann (kompatibel zur Version 2.6xx).
RunID	Laufende Nummer (RunID) der Aufgabe, deren Report bearbeitet werden soll. Format: Script-Variable oder Zahl Soll der Report einer anderen Aufgabe analysiert werden, so muss die Option "Generieren zur Laufzeit" (Registerkarte **Attribute**) aktiviert sein. Beim Report des Mandanten ist keine RunID anzugeben.

Reporttyp	Kürzel für den Typ des Reports. Format: AE-Name, Script-Literal oder Script-Variable
Filter	Vorgabe für den Zeileninhalt. Groß- und Kleinschreibung wird nicht beachtet. Für den Filter werden die Wildcardzeichen * und ? unterstützt. Bei * handelt es sich um einen Platzhalter für eine beliebige Zeichenanzahl (auch keines) und ? repräsentiert genau ein beliebiges Zeichen. Die Wildcard-Zeichen können dabei mehrfach angegeben werden. Format: Script-Literal oder Script-Variable Standardwert: "*"
Definition1	Legt fest, ob die Zeilen in Spalten unterteilt sein sollen. Format: Script-Literal oder Script-Variable Erlaubte Werte: "NONE" (Standardwert), "LENGTH", "DELIMITER" "NONE" = Keine Unterteilung in Spalten. "LENGTH" = Vorgegebene Spaltenbreiten. Erfordert LENGTH_TAB= als *Definition2*. "DELIMITER" = Spalten werden durch ein Begrenzungszeichen getrennt. Erfordert DELIMITER= als *Definition2*.

Definition2	Legt Spaltenbreiten und Spaltennamen (optional) oder das Begrenzungszeichen fest.
	Format: Script-Literal oder Script-Variable
	
	Erlaubte Werte: "LENGTH_TAB" und "DELIMITER"
	
	• "LENGTH_TAB"
	
	Spaltenbreiten und Spaltennamen (optional) werden in der Form *Spaltenbreite*=[*Spaltenname*] angegeben. Die Spaltenbreite wird über eine Zeichenanzahl definiert. Die einzelnen Spalten sind durch Kommata zu trennen (maximal 22 Spalten). Vor der ersten und nach der letzten Spaltendefinition müssen zusätzlich Hochkommata stehen. Werden für *Definition2* als Script-Literal doppelte Hochkommata verwendet, müssen das einfache Hochkommata sein, sonst umgekehrt.
	
	Beispiel:
	
	`"COL=LENGTH,LENGTH_TAB='3=drive,100=file name'"`
	
	• "DELIMITER"
	
	Das Begrenzungszeichen wird in der Form **Delimiter** angegeben.
	"*" = frei wählbares Begrenzungszeichen.
	Delimiter = Zeichenkette aus maximal 10 Zeichen, welche die Spalten voneinander trennt. Wird ein einzelnes Hochkomma als *Delimiter* verwendet, muss der gesamte Ausdruck ***COL=****Definition1[, Definition2]* in doppelte Hochkommata eingeschlossen werden und umgekehrt.
	
	Standardwert: Semikolon (;)
	
	Beispiel:
	
	`"COL=DELIMITER,DELIMITER=*'*"`
	`'COL=DELIMITER,DELIMITER=@"@'`

Rückgabewert
Referenz auf die Datensequenz des Reports.

Anmerkungen

Die Script-Funktion stellt den Reportinhalt von aktivierbaren Objekten für eine weitere Verarbeitung mit AE Script bereit. Der Report wird aus der AE Datenbank gelesen und als Datensequenz aufbereitet.

Wird auf einen Report einer Aufgabe zugegriffen, die die Scipt-Funktion aufruft, können die Parameter *RunID* und *Reporttyp* weggelassen werden. Die Script-Funktion ermittelt in diesem Fall selbst die *RunID.* Vergessen Sie jedoch nicht die zu den Parametern gehörenden Kommata zu

setzen. Standardmäßig wird bei Jobs der Jobreport ("REP") und bei anderen Aufgaben der Aktivierungsreport ("ACT") verwendet.

Der eigene Jobreport kann in der Registerkarte "Post-Script" analysiert werden. Abhängig vom Ergebnis ist es möglich, mit der Script-Anweisung :MODIFY_STATE das endgültige Ende des Jobs zu definieren.

Wird auf eine Aufgabe zugegriffen, die bereits beendet, aber deren Report noch unvollständig ist, wartet die Script-Funktion darauf.

Die Script-Funktion liest standardmäßig ganze Zeilen des Reports. Es kann aber auch strukturiert zugegriffen werden, wenn die Zeile in Spalten unterteilt ist. Hierbei gelten folgende Bestimmungen:

- maximal 22 Spalten mit einer Gesamtlänge von 2048 Bytes,
- Spaltenbreite maximal 255 Zeichen und
- Spaltenname maximal 32 Zeichen.

Mit GET_PROCESS_LINE können Sie auf einzelne Spalten zugreifen.

Der Report wird in der Sprache gespeichert, mit der das Logging der Automation Engine erfolgt.

Der Rückgabewert der Script-Funktion ist die Referenz auf eine Datensequenz. Sie wird den Script-Anweisungen :PROCESS und :ENDPROCESS als Startparameter übergeben. In Kombination mit der Script-Funktion GET_PROCESS_LINE kann damit jede einzelne Zeile der Datensequenz und deren Spalten bearbeitet werden.

Enthält der Report den gesuchten Inhalt nicht, führt das zu keiner Fehlermeldung. Die Bearbeitung der Datensequenz, definiert zwischen :PROCESS und :ENDPROCESS, wird einfach nicht durchlaufen.

Der Script-Variablen, die die Datensequenz-Referenz enthält, kann kein neuer Wert zugewiesen werden. Erst wenn Sie die Datensequenz mit der Script-Anweisung :CLOSE_PROCESS explizit verwerfen, kann die Variable erneut verwendet werden.

Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

Beispiele

Das erste Beispiel sucht alle Zeilen eines Job-Reports, in denen das Laufwerk C: vorkommt. Die kompletten Zeilen werden in das Aktivierungsprotokoll ausgegeben.

```
:SET &HND# = PREP_PROCESS_REPORT("JOBS",, "REP", "*C:\*")
:PROCESS &HND#
:    SET &RET# = GET_PROCESS_LINE(&HND#)
:    PRINT &RET#
:ENDPROCESS
```

Das zweite Beispiel basiert auf einem Job, welcher Dateinamen für eine Verarbeitung vom Benutzer abfragt hat. Die Dateinamen wurden im Aktivierungsreport gespeichert. Wenn das Hochkomma als Delimiter verwendet wird, können die Dateinamen gelesen werden. Sie werden in das Aktivierungsprotokoll ausgegeben.

```
:SET &RUNNR# = GET_UC_OBJECT_NR("MAWI.TAG")

:SET &HND# = PREP_PROCESS_REPORT(, &RUNNR#, "ACT",,"COL=DELIMITER,
DELIMITER=*'*")
:PROCESS &HND#
:    SET &RET# = GET_PROCESS_LINE(&HND#,1)
```

```
:    PRINT &RET#
:ENDPROCESS
```

Im dritten Beispiel wird in der Registerkarte **Post-Script** eines Jobs dessen eigener Job-Report ausgewertet. Dadurch kann herausgefunden werden, ob ein Fehler beim Kopieren einer Datei unter Windows auftrat. Die Script-Funktion wird lediglich mit dem Parameter *Filter* aufgerufen. Die Kommas davor stehen für den eigenen Objekttyp, die eigene laufende Nummer (RunID) und den Reporttyp REP.

```
:SET &HND# = PREP_PROCESS_REPORT(,,,"*Datei nicht finden*")
:PROCESS &HND#
:    SEND_MSG BU,BU,"Fehler beim Kopieren aufgetreten."
:    MODIFY_STATE RETCODE=50
:ENDPROCESS
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:CLOSE_PROCESS	Verwirft eine Datensequenz innerhalb eines Scripts.
:PROCESS... :TERM_ PROCESS... :ENDPROCESS	Definition einer Schleife zur zeilenweisen Bearbeitung einer Datensequenz, wie beispielsweise den Inhalt einer sequentiellen Datei oder das textmäßige Ergebnis eines Kommandos.
GET_PROCESS_ LINE	Ermittelt den aktuellen Zeileninhalt einer Datensequenz.

Script-Sprachmittel - Datensequenzen
Beispielsammlung
Endestatus aufgrund des Reportinhaltes setzen
Aufruf eines MBeans
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.9.15 PREP_PROCESS_REPORTLIST

Script-Funktion: Ermittelt die Liste des registrierten Outputs von bereits gelaufenen Jobs und stellt das Ergebnis als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.

Syntax

PREP_PROCESS_REPORTLIST(*[RunID] [, Filter]*)

Syntaxteil	Beschreibung/Format

RunID	RunID eines bereits durchgeführten Jobs, dessen registrierter Output (sowohl interne Reports als auch Job-Output-Dateien) ermittelt werden soll Format: AE-name, Script-Literal oder Script-Variable Wird der Parameter nicht angegeben, so bezieht sich das Sprachmittel auf das eigene Objekt. Das Sprachmittel sollte dann aber nur in der Registerkarte "Post Script" verwendet werden, da sonst nicht alle Job-Output Daten ermittelt werden können.
Filter	Filter auf den Namen einer Job-Output-Datei Dabei können auch die Wildcard-Zeichen * (Platzhalter beliebige Zeichen) und ? (Platzhalter für genau ein Zeichen) mehrmals verwendet werden. Format: Script-Variable oder Script-Literal

Rückgabewert
Referenz auf die Datensequenz der Liste des Job-Outputs.

Anmerkungen

Die Script-Funktion ermittelt die Liste des registrierten Job-Outputs, die in der Registerkarte "Verzeichnis" des Report-Dialoges zu finden ist. Dies schließt sowohl die externen Dateien als auch die Standard-Reports mit ein.

Der Rückgabewert der Script-Funktion ist eine Datensequenz-Referenz. Diese kann der Script-Anweisungen :PROCESS und :ENDPROCESS als Startparameter übergeben, wodurch eine Schleife entsteht. Die Anzahl der Schleifendurchläufe entspricht dabei der Anzahl der Job-Output-Einträge. Mit der Script-Funktion GET_PROCESS_LINE kann auf die einzelnen Spalten (ingesamt 8) einer jeden Zeile zugegriffen werden.

Der Script-Variablen, die die Datensequenz-Referenz enthält, kann kein neuer Wert zugewiesen werden. Erst wenn Sie die Datensequenz mit der Script-Anweisung :CLOSE_PROCESS explizit verwerfen, kann die Variable erneut verwendet werden.

Wird der Parameter *Filter* nicht angegeben, so werden alle Job-Output-Dateien verwendet (entspricht *Filter* = *).

Dabei sind folgende Spalten für jeden Eintrag auslesbar:

Spaltennummer	Wert
1	Report-Typ (zum Beispiel: ACT, REP, REV01 usw.) Externe Job-Output-Dateien besitzen folgenden Benennung: • $*NNN* - Dateien die statisch registriert wurden (Registerkarte **Output**) • #*NNN* - Dateien die dynamisch registriert wurden (Sprachmittel :REGISTER_OUTPUTFILE) NNN - dreistellige, fortlaufende Nummer
2	Zeitpunkt zu dem die Erstellung des Job-Output gestartet wurde Format: YYYY-MM-DD HH:MM:SS

3	Zeitpunkt zu dem die Erstellung des Job-Output beendet wurde Format: YYYY-MM-DD HH:MM:SS
4	Titel (nur bei SAP-Jobs)
5	Handelt es sich um einen XML-Report? (nur bei SAP-Jobs) Mögliche Werte: "0" - Nein "1" - Ja
6	Vollständiger Pfad und Name der externen Job-Output-Datei bzw. des Job-Reports (Report-Typ = REP). Bei Standard-Reports (wie ACT) ist diese Spalte leer.
7	Job-Output befindet sich am Agenten? Mögliche Werte: "0" - Nein "1" - Ja
8	Ist der Job-Output in der Datenbank gespeichert? Mögliche Werte: "0" - Nein "1" - Ja

Beispiel 1

Im ersten einfacheren Beispiel werden Name und Pfad der Output-Dateien der letzten Durchführung des Jobs JOBS.WIN.OUTPUT ausgelesen und in das Aktivierungsprotokoll geschrieben.

```
:SET &RUNID# = GET_STATISTIC_DETAIL(,RUNID,JOBS.WIN.OUTPUT)
:SET &HND# = PREP_PROCESS_REPORTLIST(&RUNID#)
:PROCESS &HND#
:SET &FILENAME# = GET_PROCESS_LINE(&HND#, 6)
: IF &FILENAME# = ""
:  ELSE
: PRINT "File name  = &FILENAME#"
: ENDIF
:ENDPROCESS
```

Beispiel 2

In folgendem Beispiel werden 2-Dateien von einem Windows-Job erzeugt und als Job-Output registriert. Folgende Zeilen befinden sich in der Registerkarte **Script** des Jobs:

```
dir C:\temp >> C:\temp\test.txt
:REGISTER_OUTPUTFILE "C:\temp\test.txt","N"
dir C:\windows >> C:\temp\test2.txt
:REGISTER_OUTPUTFILE "C:\temp\test2.txt","N"
```

In der Registerkarte **Post-Script** des Jobs wird mit dem Sprachmittel PREP_PROCESS_ REPORTLIST die komplette Job-Output-Liste abgefragt und die einzelnen Spalten jeder Zeile mit einer Process-Schleife in das Job-Protokoll geschrieben. Für jede externe Output-Datei wird ein

FileTransfer gestartet, der diese Datei auf einen anderen Rechner überträgt. Dem FileTransfer-Objekt wird dabei Agent, Login des Jobs und der vollständige Pfad der Datei übergeben.

```
:PSET &AGENT_JOB# = GET_ATT(HOST)
:PSET &LOGIN_JOB# = GET_ATT(LOGIN)

:SET &HND# = PREP_PROCESS_REPORTLIST()

:PROCESS &HND#
: SET &RH_TYPE# = GET_PROCESS_LINE(&HND#, 1)
: SET &START_TIME# = GET_PROCESS_LINE(&HND#, 2)
: SET &END_TIME# = GET_PROCESS_LINE(&HND#, 3)
: SET &TITLE# = GET_PROCESS_LINE(&HND#, 4)
: SET &IS_XML# = GET_PROCESS_LINE(&HND#, 5)
: SET &FILENAME# = GET_PROCESS_LINE(&HND#, 6)
: SET &ON_AGENT# = GET_PROCESS_LINE(&HND#, 7)
: SET &IN_DB# = GET_PROCESS_LINE(&HND#, 8)
: PRINT "Report type = &RH_TYPE#"
: PRINT "Start = &START_TIME#"
: PRINT "End = &END_TIME#"
: PRINT "Title = &TITLE#"
: PRINT "XML report? = &IS_XML#"
: PRINT "File name  = &FILENAME#"
: PRINT "On the Agent? = &ON_AGENT#"
: PRINT "In the database? = &IN_DB#"

: IF &FILENAME# = ""
:  PRINT "No external output file"
:  ELSE
: IF &ON_AGENT# = 1
: PSET &FT_FILE# = &FILENAME#
: SET &AKT# = ACTIVATE_UC_OBJECT(JOBF.OUTPUTHANDLING,,,,,PASS_VALUES,)
: ENDIF
:ENDIF

: PRINT

:ENDPROCESS
```

Im Script des FileTransfer-Objektes wird nun als Quell-Agent und Login die Daten des aktivierenden Jobs eingetragen. Zusätzlich wird auch als Quell-Datei der Pfad und Name der externen Job-Output-Datei gesetzt. Für das Ziel wird der Name der Quell-Datei übernommen, der Pfad wird auf "C:\output\" geändert.

```
:PUT_ATT FT_SRC_HOST = &AGENT_JOB#
:PUT_ATT FT_SRC_LOGIN = &LOGIN_JOB#
:PUT_ATT FT_SRC_FILE = &FT_FILE#

:SET &POS# = STR_FIND_REVERSE(&FT_FILE#, "\") + 1
:SET &FNAME# = STR_CUT(&FT_FILE#, &POS#)
:SET &DST_FILENAME# = STR_CAT("C:\output\",&FNAME#)

:PUT_ATT FT_DST_FILE = &DST_FILENAME#
```

Siehe auch:

Script-Sprachmittel	Beschreibung

PREP_PROCESS_REPORT	Ermittelt mit Filterkriterien Reportzeilen von aktivierbaren Objekten und stellt das Ergebnis als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.
:PROCESS... :TERM_PROCESS... :ENDPROCESS	Definition einer Schleife zur zeilenweisen Bearbeitung einer Datensequenz, wie beispielsweise den Inhalt einer sequentiellen Datei oder das textmäßige Ergebnis eines Kommandos.
GET_PROCESS_LINE	Ermittelt den aktuellen Zeileninhalt einer Datensequenz.

Script-Sprachmittel - Datensequenzen
Beispielsammlung
Endestatus aufgrund des Reportinhaltes setzen
Aufruf eines MBeans
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.9.16 PREP_PROCESS_VAR

Script-Funktion: Ermittelt mit Auswahlkriterien eine Liste von Werten eines Variablenobjektes und stellt diese als interne Auflistung (Datensequenz) für die weitere Verarbeitung zur Verfügung.

Syntax

PREP_PROCESS_VAR*(Variable [, Key [, Wert [, Spalte]]])*

Syntaxteil	Beschreibung/Format
Variable	Name des Variablen-Objektes, dessen Werte ausgelesen werden sollen. Format: AE-name, Script-Literal oder Script-Variable
Key	Filter für die Key-Spalte. Format: Script-Literal oder Script-Variable Maximal 64 Zeichen Standardwert: "*" Bei dynamischen Variablen ist die Key-Spalte immer die ersten Wertespalte. Die Result-Spalte wird erst nach der Ermittlung der Variableneinträge gebildet Die Wildcardzeichen "*" und "?" können verwendet werden. "*" steht dabei für beliebige, "?" für genau ein Zeichen.

Wert	Filter für den Wert. Format: Script-Literal oder Script-Variable Maximal 64 Zeichen Standardwert: "*" Die Wildcardzeichen "*" und "?" können verwendet werden. "*" steht dabei für beliebige, "?" für genau ein Zeichen.
Spalte	Werte-Spalte, auf welche sich der Filter für den Wert bezieht Format: Script-Literal, Script-Variable oder Zahl ohne Hochkommata Wildcardzeichen sind nicht erlaubt! Erlaubte Werte: Statische Variablen: "1" bis "5" Dynamische Variablen: "1" bis n Wenn Sie nur nach dem Wert filtern wollen, so ist das vorige Komma auch zu setzen. Beispiel: `:SET &HND# = PREP_PROCESS_VAR ("DATENBANKWARTUNG",,"MANDANT", 1)`

Rückgabewert
Referenz auf die Datensequenz des Variablen-Objektes.

Anmerkungen

Die Script-Funktion liest Werte eines Variablen-Objektes. Dabei können die ausgelesenen Werte durch die optionalen Parameter *Key* und *Wert* eingeschränkt werden. Bei diesen Parametern ist die Verwendung der Wildcardzeichen "*" und "?" möglich. Groß- und Kleinschreibung wird beachtet!

Wert bezieht sich auf den Inhalt einer Werte-Spalte. Mit *Spalte* legen Sie dabei die Nummer jener Werte-Spalte fest, die durchsucht werden soll. Wird *Spalte* nicht angegeben, so wird automatisch die erste Werte-Spalte verwendet. Statische Variablen besitzen 5 Werte-Spalten. Die Spalten-Anzahl von dynamischen Variablen ist nicht begrenzt und richtet sich nach der Datenquelle bzw. den Einstellungen im Variablen-Objekt.

Die Verwendung des Parameters *Spalte* ist nur in Kombination mit *Wert* möglich! Wird der Filter *Wert* nicht spezifiziert, so ist auch die Angabe einer Spalte ungültig!

Mit dem Sprachmittel ist es nicht möglich, bestimmte Spalten auszuwählen! Die gefilterten Variablen-Einträge werden mit den Werten aller Spalten, inklusive Key bzw. Result-Spalte, zurückgeliefert.

Beachten Sie, dass beim *Wert* keine Wildcardzeichen verwendet werden können, wenn es sich um eine Variable mit dem Datentyp "Zahl" handelt. Die Funktion gibt entweder alle Einträge aus oder jene die genau den angegebenen Wert aufweisen.

Beachten Sie, dass ein Leerstring "" als Filterangabe genauso wirkt wie "*", d.h. alle Werte werden zurückgeliefert.

Geben Sie *Key* und *Wert* nicht an, so werden alle Einträge des Variablen-Objektes verwendet.

Der Rückgabewert der Script-Funktion ist eine Datensequenz-Referenz. Sie wird den Script-Anweisungen :PROCESS und :ENDPROCESS als Startparameter übergeben. In Kombination mit

der Script-Funktion GET_PROCESS_LINE kann nun auf jede einzelne Zeile des Variablen-Objektes zugegriffen werden.

Enthält das Variablen-Objekt die gesuchten Werte nicht, führt das zu keiner Fehlermeldung. Die Bearbeitung der Datensequenz, definiert zwischen :PROCESS und :ENDPROCESS, wird einfach nicht durchlaufen.

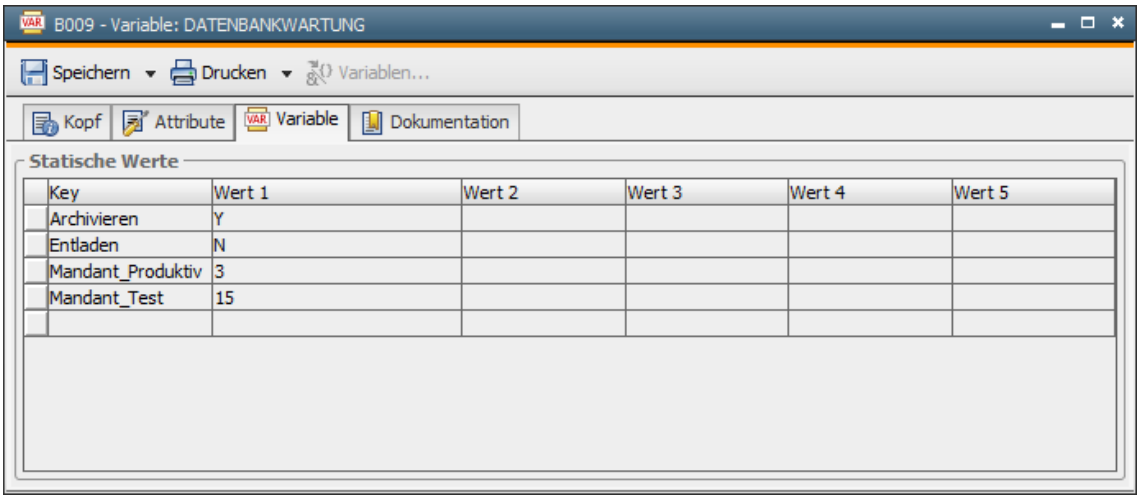
Namen von VARA-Objekten, die eine Variable enthalten, müssen in Anführungszeichen angegeben werden, ansonsten wird eine Fehlermeldung ausgegeben.

Der Script-Variablen, die die Datensequenz-Referenz enthält, kann kein neuer Wert zugewiesen werden. Erst wenn Sie die Datensequenz mit der Script-Anweisung :CLOSE_PROCESS explizit verwerfen, kann die Variable erneut verwendet werden.

Beachten Sie bei SQL-Variablen (SQL, SQLI), dass die Anzahl der zurückgelieferten Zeilen mit der Einstellung SQLVAR_MAX_ROWS (UC_SYSTEM_SETTINGS) beschränkt ist. Ist diese Einstellung beispielsweise auf den Wert 1000 gesetzt, so werden nur die ersten 1000 Zeilen zurückgeliefert.
Die zurückgelieferten Zeilen können durch den *Key*, nicht aber durch angegebenen *Wert* eingeschränkt werden. Das bedeutet: Geben Sie beim Sprachmittel PREP_PROCESS_VAR einen Filter auf einen bestimmten *Wert* an, wird nach diesem nur innerhalb der maximalen Zeilenanzahl gesucht. Wird allerdings ein *Key* angegeben, so wird dieser der Datenbank übergeben und für die Abfrage verwendet. Bei Tabellen mit vielen Einträgen ist es daher empfehlenswert, über den *Key* anstelle des *Werts* zu filtern, da ansonsten eventuell keine Einträge zurückgeliefert werden.

Beispiele

Gegeben ist das folgende Variablen-Objekt:



Key	Wert 1	Wert 2	Wert 3	Wert 4	Wert 5
Archivieren	Y				
Entladen	N				
Mandant_Produktiv	3				
Mandant_Test	15				

Im ersten Beispiel werden alle Werte ermittelt und innerhalb der Prozess-Schleife mit der :PRINT-Anweisung in das Aktivierungsprotokoll ausgegeben.

```
:SET &HND# = PREP_PROCESS_VAR(DATENBANKWARTUNG)

:PROCESS  &HND#
:   SET  &GB# = GET_PROCESS_LINE(&HND#,1)
:   SET  &WERT# = GET_PROCESS_LINE(&HND#,2)
:   PRINT "&GB# &WERT#"
:ENDPROCESS
```

```
:CLOSE_PROCESS &HND#
```

Im zweiten Beispiel werden nur jene Einträge gelesen, deren Key mit "Mandant" beginnt.

```
:SET  &HND# = PREP_PROCESS_VAR(DATENBANKWARTUNG,"Mandant*")

:PROCESS  &HND#
:   SET  &GB# = GET_PROCESS_LINE(&HND#,1)
:   SET  &WERT# = GET_PROCESS_LINE(&HND#,2)
:   PRINT "&GB# &WERT#"
:ENDPROCESS

:CLOSE_PROCESS &HND#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:CLOSE_PROCESS	Verwirft eine Datensequenz innerhalb eines Scripts.
:PROCESS... :TERM_ PROCESS... :ENDPROCESS	Definition einer Schleife zur zeilenweisen Bearbeitung einer Datensequenz, wie beispielsweise den Inhalt einer sequentiellen Datei oder das textmäßige Ergebnis eines Kommandos.
GET_PROCESS_ LINE	Ermittelt den aktuellen Zeileninhalt einer Datensequenz.

Script-Sprachmittel - Datensequenzen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.9.17 PUT_PROCESS_LINE

Script-Funktion: Fügt einer bestimmten Datensequenz eine Zeile hinzu.

Syntax

PUT_PROCESS_LINE*(Datensequenz-Referenz, Zeile [, Trennzeichen])*

Syntaxteil	Beschreibung/Format
Datensequenz-Referenz	Referenz auf die Datensequenz, der eine Zeile hinzugefügt werden soll. Format: Script-Variable
Zeile	Zeile, welche der Datensequenz angehängt wird. Format: Script-Variable oder Script-Literal Auch die Angabe eines Script Arrays ist möglich.
Trennzeichen	Beliebiges Zeichen, welches die angegebene Zeile in Spalten aufteilt. Format: Script-Variable oder Script-Literal Wird für *Zeile* ein Array angegeben, besitzt dieser Parameter keine Wirkung.

Rückgabewert
"0" - Datensequenz wurde erfolgreich um die angegebene Zeile erweitert.

Anmerkungen

Die Script-Funktion fügt einer Datensequenz eine neue Zeile hinzu. Die Datensequenz wird dabei direkt geändert - der Rückgabewert der Funktion zeigt nur, ob das Hinzufügen der Zeile erfolgreich durchgeführt werden konnte. Dabei ist die Referenz auf die zu ändernde Datensequenz und der Wert, der als neue Zeile eingefügt werden soll, anzugeben.

Datensequenzen können mit dem Sprachmittel CREATE_PROCESS oder mit den PREP_PROCESS* - Funktionen erstellt werden.

Achten Sie darauf, dass die Datensequenz, deren Referenz angegeben wurde, tatsächlich existiert oder nicht bereits via :CLOSE_PROCESS geschlossen wurde. Ist dies nicht der Fall, kommt es zu einem Laufzeitfehler. Mit dem Sprachmittel kann keine neue Datensequenz erstellt werden!

Die hinzuzufügende *Zeile* kann dabei entweder in Form einer Zeichenkette (Script-Literal bzw. Script-Variable) oder eines Script-Array (Script-Variable) angegeben werden. Bei einer Zeichenkette kann zusätzlich ein *Trennzeichen* angegeben werden, gemäß diesem die Aufteilung in mehrere Spalten erfolgt. Wird kein Trennzeichen spezifiziert, so besitzt die neue Zeile nur eine Spalte, die den kompletten Wert enthält.
Wird ein Array angegeben, so stellen dessen Elemente die Spalten der neuen Zeile dar. Die leeren Elemente am Ende des Arrays werden nicht verwendet.

Beispiele

Im folgendem Beispiel wird eine Datensequenz erstellt, welche eine Dateiliste speichert. Anschließend wird ein Dateiname aus einem Variablen-Objekt gelesen und der Datensequenz hinzugefügt.

```
:SET &HND# = PREP_PROCESS_FILENAME
("WIN01","C:\AUTOMIC\temp\test*.txt","Y",,)
:SET &LINE# = GET_VAR(VARA.FILELIST)
:SET &RET# = PUT_PROCESS_LINE(&HND#,&LINE#)
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:CLOSE_PROCESS	Verwirft eine Datensequenz innerhalb eines Scripts.
:PROCESS... :TERM_ PROCESS... :ENDPROCESS	Definition einer Schleife zur zeilenweisen Bearbeitung einer Datensequenz, wie beispielsweise den Inhalt einer sequentiellen Datei oder das textmäßige Ergebnis eines Kommandos.
CREATE_ PROCESS	Erstellt ein neue Datensequenz.
GET_PROCESS_ LINE	Ermittelt den aktuellen Zeileninhalt einer Datensequenz.

Script-Sprachmittel - Datensequenzen

Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.9.18 SAVE_PROCESS

Script-Funktion: Speichert eine bestimmte Datensequenz.

Syntax

SAVE_PROCESS*(Datensequenz-Referenz)*

Syntaxteil	Beschreibung/Format
Datensequenz	Referenz auf die Datensequenz, die gespeichert werden soll. Format: Script-Variable

Rückgabewert
Begriff, unter dem die Datensequenz abgelegt wird.

Anmerkungen

Die Script-Funktion speichert eine bestimmte Datensequenz für die Aktivität in der Datenbank. Dadurch können andere Aufgaben auf diese Datensequenz zugreifen.

Als Rückgabewert wird ein Begriff zurückgeliefert, der sich aus dem Namen der Datensequenz-Referenz zusammensetzt. Um gespeicherte Datensequenzen in anderen Aufgaben verwenden zu können, sind diese zuerst mit dem Sprachmittel LOAD_PROCESS unter Angabe dieses Begriffs zu laden.

Gespeicherte Datensequenzen können beliebig oft von beliebigen Aufgaben geladen werden.

Beachten Sie, dass SAVE_PROCESS die Datensequenz zwar speichert, jedoch nicht automatisch schließt. Die Verwendung des Sprachmittels CLOSE_PROCESS ist daher nach wie vor notwendig!

Die Datensequenzen bleiben bis zur Reorganisation der Statistiksätze jener Aufgaben erhalten, über welche sie gespeichert wurden.

Beispiel

Zwei Script-Objekte SCRI.TEST.DS1 und SCRI.TEST.DS2 werden in einem Workflow hintereinander ausgeführt.

Die erste Aufgabe SCRI.TEST.DS1 erstellt eine neue Datensequenz und speichert diese in die Datenbank. Die RunID der Aufgabe und der interne Name der gespeicherten Datensequenz werden dem Workflow via Objektvariablen zur Verfügung gestellt. Zum Abschluss wird die Datensequenz geschlossen. Dies wird mit folgendem Script bewerkstelligt:

```
:SET &HND# = PREP_PROCESS_VAR(VARA.DB)
:PSET &HND_KEY# = SAVE_PROCESS(&HND#)
```

```
:PSET &RUNID# = SYS_ACT_ME_NR()
:CLOSE_PROCESS &HND#
```

Die zweite Aufgabe SCRI.TEST.DS2 erbt die Objektvariablen des Workflows. Mit Hilfe dieser Informationen wird die Datensequenz geladen, deren Inhalt ausgelesen und im Aktivierungsprotokoll ausgegeben.

```
:SET &HND# = LOAD_PROCESS(&RUNID#,&HND_KEY#)
:PROCESS &HND#
:  SET &LINE# = GET_PROCESS_LINE(&HND#,2)
: P &LINE#
:ENDPROCESS
:CLOSE_PROCESS &HND#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:CLOSE_PROCESS	Verwirft eine Datensequenz innerhalb eines Scripts.
:PROCESS... :TERM_ PROCESS... :ENDPROCESS	Definition einer Schleife zur zeilenweisen Bearbeitung einer Datensequenz, wie beispielsweise den Inhalt einer sequentiellen Datei oder das textmäßige Ergebnis eines Kommandos.
CREATE_ PROCESS	Erstellt ein neue Datensequenz.
GET_PROCESS_ LINE	Ermittelt den aktuellen Zeileninhalt einer Datensequenz.
LOAD_PROCESS	Lädt eine gespeicherte Datensequenz.

Script-Sprachmittel - Datensequenzen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.9.19 WRITE_PROCESS

Script-Funktion: Schreibt den Inhalt einer Datensequenz in eine Datei.

Syntax

WRITE_PROCESS*(Datensequenz-Referenz, Datei, Agent, Login [,[Modus] [,[CodeTabelle] [,[Dateiattribute] [,Spalte]]]])*
WRITE_PROCESS*(Datensequenz-Referenz, Datei, Agent, Login [,[Modus] [,[CodeTabelle] [,[Dateiattribute] [,***ALL** *[,Delimiter]]]]])*

Syntaxteil	Beschreibung/Format
Datensequenz-Referenz	Referenz auf jene Datensequenz, deren Inhalt in eine Datei exportiert werden soll. Format: Script-Variable

Datei	Pfad und Name der Datei Format: Script-Variable oder Script-Literal
Agent	Name des Agenten, auf dessen Host die Datei gespeichert wird. Format: AE-Name, Script-Variable oder Script-Variable
Login	Name des Login-Objektes, welches für die Anmeldung am Host des angegebenen Agenten verwendet wird. Format: AE-Name, Script-Variable oder Script-Variable
Modus	Verhalten, wenn die angegebene Datei bereits existiert. Format: AE-Name, Script-Variable oder Script-Variable Erlaubte Werte: CANCEL (Standardwert)- Script-Durchführung abbrechen OVERWRITE - Datei überschreiben APPEND - bestehende Datei um die neuen Zeilen erweitern
CodeTabelle	Name eines CodeTabellen-Objektes, welches für die Kodierung der Datei verwendet werden soll. Format: AE-Name, Script-Variable oder Script-Variable Wird dieser Parameter nicht angegeben, so wird die Standard-CodeTabelle UC_CODE verwendet.
Dateiattribute	Zusätzliche Dateiattribute für die erzeugte Datei Format: Script-Variable oder Script-Literal Abhängig von der Plattform des Agenten können Sie jene Attribute angeben, welche auch für FileTransfers spezifiziert werden können. Mehrere Attribute sind dabei durch Kommata zu trennen.
Spalte	Spalte(n) der Datensequenz, welche in die Datei geschrieben werden sollen. Format: Script-Literal, Script-Variable, Zahl ohne Hochkommata Erlaubte Werte: ALL (Standardwert) - Alle Spalten Nummer der entsprechenden Spalte (z.B.: 2)
Delimiter	Trennzeichen bei der Angabe mehrerer Spalten Format: Script-Literal, Script-Variable, Zahl ohne Hochkommata Nur relevant, wenn alle Spalten der Datensequenz exportiert werden.

Rückgabewert
0 - Datei erfolgreich exportiert 20554 - Agent derzeit nicht aktiv

Anmerkungen

Das Sprachmittel exportiert den Inhalt einer beliebigen Datensequenz in eine Text-Datei. Anzugeben ist dabei die Referenz auf die entsprechende Datensequenz, welche beim Anlegen der Datensequenz in einer Script-Variable abgelegt wird.

Jede Zeile der Datensequenz wird als Zeile in die Datei geschrieben.

Es können entweder alle oder nur eine bestimmte Spalte der Datensequenz in die Datei geschrieben. Dies ist mit dem Parameter Spalte festzulegen, wobei standardmäßig alle Spalten exportiert werden.

⚠ Der Parameter *Delimiter* ist nur relevant, wenn alle Spalten exportiert werden. Er legt das Zeichen fest, gemäß welchem die Datensequenz-Zeilen in Spalten aufgetrennt werden.

Beispiele

Folgendes Beispiel erstellt eine neue Datensequenz, welche eine Liste mit Dateinamen eines bestimmten Verzeichnisses speichert. Anschließend wird der Inhalt dieser Datensequenz in eine Datei auf einem anderen Host abgelegt. Kommt es dabei zu einem Fehler (Rückgabewert von WRITE_PROCESS > 0), so wird eine Fehlermeldung erstellt und an den betreffenden AE-Benutzer gesendet.

```
:SET &HND# = PREP_PROCESS_FILENAME
(
"WIN01"
,"c:\AUTOMIC\server\bin\*.exe",,,,"COL=DELIMITER","DELIMITER=*\*")
:SET &RET# = WRITE_PROCESS(&HND#,"C:\temp\export.txt",WIN02,
LOGIN.GLOBAL,OVERWRITE)

:IF &RET# > 0
: SET &ERRNR# = SYS_LAST_ERR_NR()
: SET &ERRINS# = SYS_LAST_ERR_INS()
: SET &MESSAGE# = GET_MSG_TXT(&ERRNR#,&ERRINS#)
: SEND_MSG &$USER#, &$DEPARTMENT#, &MESSAGE#
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
CREATE_PROCESS	Erstellt ein neue Datensequenz.
GET_PROCESS_LINE	Ermittelt den aktuellen Zeileninhalt einer Datensequenz.
LOAD_PROCESS	Lädt eine gespeicherte Datensequenz.
PREP_PROCESS	Führt mit Hilfe von speziellen Job-Objekten (Ereignis-Jobs), Kommandos auf einem Rechner aus, und liefert die Konsolen-Ausgabe als interne Auflistung (Datensequenz) zurück, die zur Weiterverarbeitung genutzt werden kann.

Script-Sprachmittel - Datensequenzen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.10 Ereignisbehandlung

3.10.1 GET_CONSOLE, GET_EVENT_INFO

Script-Funktion: Liest Daten von eingetretenen Konsolen-, Dateisystem- und Datenbank-Ereignissen aus.

Syntax

GET_CONSOLE*(Schlüsselwort*[, *Index*]*)*
GET_EVENT_INFO*(Schlüsselwort*[, *Index*]*)*

Syntaxteil	Beschreibung/Format
Schlüsselwort	Gibt an, welcher Bestandteil der Meldung gelesen werden soll. Format: AE-name, Script-Literal oder Script-Variable Die Schlüsselwörter für die einzelnen Plattformen werden in den unten stehenden Tabellen erläutert.
Index	Kann nur in Verbindung mit den Schlüsselwörtern INSERT, RESULT1 und RESULT2 verwendet werden. Format: Script-Literal, Zahl oder Script-Variable. Standardwert: 1 Für INSERT: Zugriff auf die 16 Bit-Felder einer Konsolen-Meldung in z/OS und Zugriff auf den variablen Teil des Nachrichtentextes (Insert) einer OS/400-Nachricht. Für RESULT1 und RESULT2: Nummer der Spalte der SQL Abfrage (Datenbank-Ereignis)

Schlüsselwörter

Schlüsselwort	Konsole BS2000	Konsole z/OS	Konsole OS/400	Konsole SAP (ABAP)	Konsole SAP (XI)	Konsole SAP (Java)	Konsole Windows	Dateisystem	Dateisystem automatisch z/OS	Datenbank
ACCESS									✓	
ACTIVATIONSTATE					✓					
ALERT_OPTION			✓							
CATEGORY							✓			
CHANNEL					✓					

DDNAME									✓	
EVENT_ ADDITIONA L_PARM						✓				
EVENT_ COUNTOFJ OBS				✓						
EVENT_ GUID				✓						
EVENT_ID				✓		✓	✓			
EVENT_ PARM				✓		✓				
EVENT_ PROCESSS TATE				✓						
EVENT_ SERVER				✓						
EVENT_ STATE				✓						
EVENT_ TYPE						✓				
FILENAME								✓	✓	
FILESIZE								✓		
ID					✓					
INFO_ ACTION									✓	
INFO_TEXT									✓	
INSERT			✓				✓			
INSERT_ COUNT			✓				✓			
JOB_ ABENDED									✓	
JOB_ ENDED									✓	
JOB_ID		✓							✓	
JOB_NAME		✓							✓	
LPAR_ NAME		✓							✓	
MEMGEN									✓	

MSG_ DESCRIPT OR		✓							
MSG_FILE_ LIBRARY			✓						
MSG_ FILENAME			✓						
MSG_ID			✓						
MSG_KEY			✓						
MSG_LEVEL		✓							
MSG_ LIBRARY_ USED			✓						
MSG_MISC		✓							
MSG_ SEVERITY			✓						
MSG_TEXT	✓	✓	✓						
MSGNR	✓								
MSGTYPE, MSGTYP	✓		✓						
OS_NAME		✓							
PARTY					✓				
PROCESSI D	✓								
PROGRAM_ NAME								✓	
REPLY_ID		✓							
RESULT							✓		
RESULT1									✓
RESULT2									✓
RETCODE								✓	
RETCODE_ STATE								✓	
SEND_JOB			✓						
SEND_JOB_ NUMBER			✓						
SEND_ USER_ PROFILE			✓						

SERVICE					✓					
SMS_ MANAGEMENT_CLASS SMS_ DATA_ CLASS SMS_ STORAGE_ CLASS									✓	
SOURCE							✓			
STATE					✓					
STEP_ NAME									✓	
SYS_NAME		✓							✓	
SYSPLEX_ NAME		✓							✓	
TIMESTAMP	✓	✓	✓	✓		✓	✓	✓	✓	
TYPE							✓			
USER							✓			

⚠ Bitte beachten Sie, dass die Verwendung der Schlüsselwörter, die in der Spalte "Dateisystem" aufgelistet sind, nur bei Dateisystem-Ereignissen mit der Einstellung für "Prüfung" gleich "FILE_SIZE" bzw. "leer" (mit oder ohne Kombination von "FILE_STABLE" oder "FILE_CHANGED") und "PATH_FILE_COUNT" (jedoch nur in Kombination mit "FILE_STABLE" oder "FILE_CHANGED") möglich ist.

Beschreibung

Schlüsselwort	Beschreibung
ACCESS	Zugriff auf die Datei. Erlaubter Wert: "CLOSE" - Schließen der Datei
ACTIVATIONSTATE	Status des Adaptors Erlaubte Werte: "STARTED" und "STOPPED"

ALERT_OPTION	Gibt zurück, ob und wann ein SNA-Warnsignal erzeugt und zur Nachricht geschickt wird. Erlaubte Werte: "*DEFER" - Das Warnsignal wird nach der lokalen Problemanalyse geschickt. "*IMMED" - Es wird sofort ein Warnsignal geschickt, gleichzeitig mit dem Senden der Nachricht. Für die Nachrichtenwarteschlange muss das Attribut, Alarm zu erlauben, auf "*YES" gesetzt sein. "*NO" - Es wird kein Warnsignal geschickt. "*UNATTEND" - Ein Warnsignal wird nur dann geschickt, wenn das System im nicht überwachten Modus läuft. Dazu muss das Alarmstatus Netzwerkattribut (ALRSTS) "*UNATTEND" sein.
CATEGORY	Kategorie des Ereignisses
CHANNEL	Kommunikationskanal
DDNAME	DD-Bezeichnung
EVENT_ ADDITIONAL_PARM	Zusätzliche Parameter von SAP Java Scheduler Ereignissen.
EVENT_ COUNTOFJOBS	Anzahl an SAP-Jobs, die angestoßen wurden (ABAP)
EVENT_GUID	Instanznummer des SAP-Ereignisses (ABAP)
EVENT_ID	Ereignis-ID (ABAP /Java Ereignisse)
EVENT_PARM	SAP-Ereignisparameter (ABAP/Java Ereignisse)
EVENT_ PROCESSSTATE	SAP-Verarbeitungsstatus (ABAP)
EVENT_SERVER	SAP-Ereignisserver (ABAP)
EVENT_STATE	SAP-Ereignisstatus(ABAP)
EVENT_TYPE	SAP Java Scheduler Ereignistyp.
FILENAME	Name der Datei Bei Dateisystem-Ereignissen mit Wildcards wird folgender Wert als Dateiname zurückgeliefert: • *ALL - Wenn die Option "Alle Dateien berücksichtigen" aktiviert ist. • *ONE - Wenn die Option "Alle Dateien berücksichtigen" deaktiviert ist.
FILESIZE	Größe der Datei
ID	ID der Nachricht
INFO_ACTION	Aktion Erlaubte Werte: "Y" und "N" "Y" - Ein Objekt wurde aktiviert. "N" - Es erfolgte keine Aktivierung eines Objektes.
INFO_TEXT	Name des Objektes, das aktiviert wurde.
INSERT	Greift zusammen mit *Index* auf einen bestimmten variablen Nachrichtenteil zu.

INSERT_COUNT	Ermittelt die Anzahl der variablen Nachrichtenteile.
JOB_ABENDED	Endestatus des Jobs, der die Datei geschlossen hat Erlaubte Werte: "Y" und "N" "Y" - Der Job endete abnormal. "N" - Der Job endete normal.
JOB_ENDED	Status des Jobs, der die Datei geschlossen hat Erlaubte Werte: "Y" und "N" "Y" - Der Job ist schon beendet. "N" - Der Job läuft noch.
JOB_ID	Identifikationsbezeichnung des Jobs
JOB_NAME	Name des Jobs
LPAR_NAME	Gibt den LPAR-Namen zurück
MEMGEN	Membername oder Generation-Number
MSG_DESCRIPTOR	Beschreibung der Meldung als 16 Bit-Feld. Bit 01(entspricht X'80') = SYSTEM_FAILURE - System failure Bit 02 (entspricht X'40') = IMMEDIATE_ACTION - Immediate Action Required Bit 03 (entspricht X'20') = EVENTUAL_ACTION - Eventual Action Required Bit 04 (entspricht X'10') = SYSTEM_STATUS - System status Bit 05 (entspricht X'08') = IMMEDIATE_COMMAND - Immediate command response Bit 06 (entspricht X'04') = JOB_STATUS - Job status Bit 07 (entspricht X'02') = APPLICATION - Application program/processor Bit 08 (entspricht X'01') = OUT_OF_LINE - Out-of-line Bit 09 (entspricht X'80') = OPERATOR_REQUEST - Operator's request Bit 10 (entspricht X'40') = TRACK_COMMAND_R - Track command response Bit 11 (entspricht X'20') = CRITICAL_ACTION - critical eventual action Bit 12 (entspricht X'10') = IMPORTANT_INFO - important information Bit 13 (entspricht X'08') = PREVIOUSLY_AUTO - previously automated Bit 14 (entspricht X'04') - reserviert Bit 15 (entspricht X'02') - reserviert Bit 16 (entspricht X'01') - reserviert
MSG_FILE_ LIBRARY	Name der Bibliothek, welche die Nachrichtendatei enthält oder die vom Sendeprogramm verwendeten Werte "*CURLIB" bzw. "*LIBL".
MSG_FILENAME	Name der Nachrichtendatei, welche die erhaltene Nachricht enthält.
MSG_ID	Nachrichten-ID der erhaltenen Nachricht. Bei einer improvisierten Nachricht, wird ein Leerzeichen zurückgegeben.
MSG_KEY	Nachrichtenschlüssel der erhaltenen Nachricht. 4-stelliger Wert im hexadezimalen Format X'xxxxxxxx'.

MSG_LEVEL	Dringlichkeitsstufe der Meldung als 16 Bit-Feld. Bit 01(entspricht X'80') = WTOR - WTOR Bit 02 (entspricht X'40') = IMMEDIATE_ACTION - Immediate Action Bit 03 (entspricht X'20') = CRITICAL_ACTION - Critical Eventual Action Bit 04 (entspricht X'10') = EVENTUAL_ACTION - Eventual Action Required Bit 05 (entspricht X'08') = INFO - Informational Bit 06 (entspricht X'04') = BROADCAST Bit 07 (entspricht X'02') - reserviert Bit 08 (entspricht X'01') - reserviert Bit 09 (entspricht X'80') - reserviert Bit 10 (entspricht X'40') - reserviert Bit 11 (entspricht X'20') - reserviert Bit 12 (entspricht X'10') - reserviert Bit 13 (entspricht X'08') - reserviert Bit 14 (entspricht X'04') - reserviert Bit 15 (entspricht X'02') - reserviert Bit 16 (entspricht X'01') - reserviert
MSG_LIBRARY_ USED	Tatsächlicher Name der Bibliothek, welche verwendet wurde, um die Nachricht zu senden. Weil die Bibliothek Override-Anweisungen enthalten kann, ist es nicht notwendigerweise die Bibliothek, in der sich die Nachricht gegenwärtig befindet.
MSG_MISC	Verschiedene Zusatzinformationen zur Meldung als 16 Bit-Feld. Bit 01(entspricht X'80') = DISPLAY_UD_MSG - display UD messages Bit 02 (entspricht X'40') = DISPLAY_ONLY_UD_MSG - display only UD messages Bit 03 (entspricht X'20') = QUEUE_BY_ID_ONLY - queue by id only Bit 04 (entspricht X'10') = QUEUE_BY_AUTO - queue by automation Bit 05 (entspricht X'08') = QUEUE_BY_HARDCOPY - queue by hardcopy Bit 06 (entspricht X'04') - reserviert Bit 07 (entspricht X'02') - reserviert Bit 08 (entspricht X'01') - reserviert Bit 09 (entspricht X'80') = ECHO_OPERATOR_CMD - echo Operator command Bit 10 (entspricht X'40') = ECHO_INTERNAL_CMD - echo internal command Bit 11 (entspricht X'20') = RESULT_OF_WTL_MACRO - result of wtl macro Bit 12 (entspricht X'10') - reserviert Bit 13 (entspricht X'08') - reserviert Bit 14 (entspricht X'04') - reserviert Bit 15 (entspricht X'02') - reserviert Bit 16 (entspricht X'01') - reserviert
MSG_SEVERITY	Bewertung der erhaltenen Nachricht. Erlaubte Werte: "0" bis "99".
MSG_TEXT	Meldungstext (Defaultwert)
MSGNR	Meldungsnummer

MSGTYPE, MSGTYP	Meldungstyp Erlaubte Werte speziell bei OS/400: "01" - Beendigung "02" - Diagnose "04" - Information "05" - Anfrage "06" - Kopie "08" - Anforderung "10" - Anforderung mit Bedienerführung "14" - Hinweisnachricht (Ausnahmebedingung bereits behandelt) "15" - Abbruchnachricht (Ausnahmebedingung bereits behandelt) "16" - Hinweisnachricht (Ausnahmebedingung noch nicht behandelt) "17" - Abbruchnachricht (Ausnahmebedingung noch nicht behandelt) "21" - Antwort, Gültigkeit nicht geprüft "22" - Antwort, Gültigkeit geprüft "23" - Antwort, Standardnachricht "24" - Antwort, Standardantwort des Systems "25" - Antwort, aus der Antwortliste des Systems
OS_NAME	Von IBM definierter Name des Betriebssystems in der Konsolen-Meldung. Derzeit immer "MVS".
PARTY	Partner
PROCESSID	Tasknummer (TSN) in der Konsolen-Meldung.
PROGRAM_NAME	Progammbezeichnung
REPLY_ID	Reply ID der Konsolen-Meldung.
RETCODE	Rückgabewert des Jobs im umgewandelten Format
RETCODE_STATE	Filterung auf den Rückgabewert Erlaubte Werte: "Y" und "N" "Y" - Es wurde ein Filter für den Rückgabewert gesetzt. "N" - Der Rückgabewert ist für die Auslösung des Ereignisses nicht relevant.
RESULT	Zeigt an, ob bei Dateisystem-Ereignissen mit den Prüfungen FILESIZE oder PATH_FILE_COUNT die Bedingung erfüllt worden ist. Erlaubte Werte: "Y" und "N" "Y" - Die Bedingung wurde erfüllt. "N" - Die Bedingung wurde nicht erfüllt.
RESULT1 RESULT2	RESULT1 erlaubt den Zugriff auf das Ergebnis der SQL Abfrage von "Wert 1", RESULT2 auf jenes von "Wert 2". Geben Sie beim Index die Nummer der Spalte an dessen Wert ausgelesen werden soll. ⚠ Beachten Sie, dass das Datenbank-Ereignis nur die ersten 10 Spalten ausliest. Außerdem werden Spalteninhalte mit mehr als 255 Zeichen abgeschnitten. Die Script-Funktion liefert den Rückgabewert " ", wenn Sie auf eine Spalte zugreifen, die nicht existiert.
SEND_JOB	Name des Jobs, in dem die erhaltene Nachricht gesendet wurde.

SEND_JOB_ NUMBER	Nummer des Jobs, in dem die erhaltene Nachricht gesendet wurde.
SEND_USER_ PROFILE	Name des Benutzerprofils, welches die erhaltene Nachricht gesendet hat.
SERVICE	Service
SMS_ MANAGEMENT_ CLASS SMS_DATA_CLASS SMS_STORAGE_ CLASS	Name der Klasse
SOURCE	Quelle des Ereignisses
STATE	Zustand Erlaubte Werte: "*" "ERROR" "OK" "INACTIVE" "UNKNOWN" "UNREGISTERED"
STEP_NAME	Name des Jobsteps
SYS_NAME	Benutzerdefinierter System-Name
SYSPLEX_NAME	SYSPLEX-Name
TIMESTAMP	Datum und Uhrzeit der Konsolen-Meldung.
TYPE	Art des Ereignisses Erlaubte Werte: "I" - Information "W" - Warnung "E" - Fehler "S" - Überwachung-Erfolgreich "F"- Überwachung-Fehlversuch
USER	Benutzer

Anmerkungen

Die Script-Sprachmittel GET_CONSOLE und GET_EVENT_INFO bieten Ihnen die Möglichkeit Informationen zum eingetretenen Ereignis zu ermitteln. Die Syntax ist bei beiden gleich.

Die Script-Funktion GET_CONSOLE erlaubt es, beim Eintreten eines Konsolen-Ereignisses Meldungsdaten zu ermitteln. Das sind definierte Bestandteile der Konsolen-Meldung, welche mit *Schlüsselwort* angegeben werden können. Als Standardwert liefert die Funktion den Meldungstext.

Die Script-Funktion kann in einem Ereignis vom Typ "Konsole" verwendet werden. Im BS2000 und z/OS kann damit die Konsolen-Meldung gelesen werden. Im OS/400 dient die Script-Funktion dazu, Informationen aus einer Nachrichtenwarteschlange zu erhalten. Welche Nachrichtenwarteschlange überwacht werden soll, kann in der INI-Datei des OS/400-Agenten festgelegt werden.

Mit GET_EVENT_INFO können Sie im Script von Dateisystem-Ereignissen Informationen auslesen. Speziell bei z/OS stehen Ihnen dazu mehrere Schlüsselwörter zur Verfügung.

Plattformspezifische Besonderheiten

z/OS

Die Schlüsselworte MSG_DESCRIPTOR, MSG_LEVEL und MSG_MISC stellen eine Besonderheit dar. Es sind 16 Bit-Felder, bei denen jedes einzelne Bit eine bestimmte Bedeutung hat. Mit der Script-Funktion kann jedes einzelne Bit abgefragt werden. Dazu wird mit *Index* das Bit als Wert oder als Konstante angegeben. Der Rückgabewert der Script-Funktion ist "1" (Bit ist gesetzt) oder "0" (Bit ist nicht gesetzt).

OS/400

Mit INSERT_COUNT als *Schlüsselwort* kann die Anzahl variabler Nachrichtenteile einer OS/400-Nachricht festgestellt werden. Mit INSERT als *Schlüsselwort* und der Angabe von *Index* wird auf einen bestimmten variablen Nachrichtenteil zugegriffen. *Index* kann ausschließlich zu INSERT optional angegeben werden. Ohne *Index* gibt die Funktion den ersten variablen Nachrichtenteil zurück.

Beispiele

Die Script-Funktion GET_CONSOLE ermittelt im Beispiel die TSN des Prozesses, welcher das Ereignis auslöste.

```
:SET &TSN#=GET_CONSOLE(PROCESSID)
```

Im zweiten Beispiel wird als erstes die Anzahl der variablen Teile des Nachrichtentextes (Inserts) einer OS/400-Nachricht ermittelt. Danach wird eine Prozess-Schleife durchlaufen, in der alle Inserts in den Aktivierungsreport geschrieben werden.

```
:SET &COUNT# = GET_CONSOLE("INSERT_COUNT")
:SET &IDX# = 1
:WHILE &COUNT# > 0
:   SET &INSERT# = GET_CONSOLE("INSERT", &IDX#)
:   SET &HELP# = FORMAT(&IDX#, "000")
:   PRINT "INSERT[&HELP#] = '&INSERT#'"
:   SET &IDX# = ADD(&IDX#, 1)
:   SET &COUNT# = SUB(&COUNT#, 1)
:ENDWHILE
```

Folgende Zeilen werden im Aktivierungsreport protokolliert:

```
20010110/235011.000 - U0020408 INSERT[001] = 'QPFRMON'
20010110/235011.000 - U0020408 INSERT[002] = 'QPGMR'
20010110/235011.000 - U0020408 INSERT[003] = '007982'
20010110/235011.000 - U0020408 INSERT[004] = '23:48:43'
20010110/235011.000 - U0020408 INSERT[005] = '10/01/01'
20010110/235011.000 - U0020408 INSERT[006] = '0'
```

Das dritte Beispiel ermittelt einen Bestandteil der Konsolen-Meldung in z/OS. Es wird auf Bit 03 des Schlüsselwortes MSG_DESCRIPTOR zugegriffen. Bit 03 wird zum einen als Wert, zum anderen als Konstante angegeben.

```
:SET &RET# = GET_CONSOLE("MSG_DESCRIPTOR", 3)
:SET &RET# = GET_CONSOLE("MSG_DESCRIPTOR", "EVENTUAL_ACTION")
```

Das folgende Beispiel zeigt einen Scriptauszug in dem der Dateiname ermittelt und der Dateiinhalt zeilenweise eingelesen wird.

```
:SET &DATEINAME# = GET_EVENT_INFO (FILENAME)
:SET &HND# = PREP_PROCESS_FILE ("MVSHOST", &DATEINAME#)
```

Das fünfte Beispiel liest die Werte einer SQL Abfrage (für "Wert 1") in einem Datenbank-Ereignis aus.

SQL-Ergebnisses:

Nachname	Vorname	Ort
Meier	Hans	Wolfsgraben

Der Vorname wird wie folgt ermittelt:

```
:SET &VORNAME# = GET_EVENT_INFO (RESULT1, 2)
```

Die folgende Zeile liefert den Ort:

```
:SET &ORT# = GET_EVENT_INFO (RESULT1, 3)
```

Siehe auch:

Script-Sprachmittel	Beschreibung
GET_BIT	Prüft, ob in einem Bit-Feld ein bestimmtes Bit gesetzt ist.

Script-Sprachmittel - Ereignisbehandlung
Beispielsammlung
Reaktion auf externe Ereignisse
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.10.2 GET_FILESYSTEM

Script-Funktion: Ermittelt von einem Rechner ab einem definierten Pfad verschiedene Dateisystemwerte.

Syntax

GET_FILESYSTEM*([Host],[Pfad],**Dateisystem-Wert**[,Maßeinheit] [,Unterverzeichnisse einbeziehen])*

Syntaxteil	Beschreibung/Format

Host	Name des Agenten, der auf dem Rechner läuft von dem Informationen ermittelt werden sollen. Format: AE-name, Script-Literal oder Script-Variable Geben Sie den Hostnamen nicht an, so wird der zuletzt mit der Script-Funktion GET_FILESYSTEM benutzte Agent verwendet. Die Werte werden nicht neu ermittelt, sondern nur jene des letzten Aufrufs ausgegeben.
Pfad	Angabe zu Dateien oder Dateisystemen, zu denen Informationen ermittelt werden sollen. Hier können, abhängig vom jeweiligen Zielsystem, Dateinamen, Laufwerke, Volumes, Pfade, Kennungen, Generation Data Groups (GDG) usw. qualifiziert oder teilqualifiziert benannt werden. Es können die Wildcardzeichen "*" oder "?" verwendet werden. "*" steht dabei für beliebige, "?" für genau ein Zeichen. Beachten Sie, dass bei Windows die Wildcardzeichen "*" und "?" nur für den Dateinamen, nicht aber für Verzeichnisse innerhalb des Pfads verwendet werden können! Die Zeichen * und ? werden immer als Wildcards angewendet. Es ist daher nicht möglich, gezielt Dateien anzugeben, welche diese Zeichen im Namen aufweisen. Format: AE-Name, Script-Literal oder Script-Variable Geben Sie den Pfadnamen nicht an, so wird der zuletzt mit der Script-Funktion GET_FILESYSTEM benutzte Pfadname verwendet. Die Werte werden nicht neu ermittelt, sondern nur jene des letzten Aufrufs ausgegeben. Schließen Sie die Pfadangaben auf Windows unbedingt mit einem "*" ab. Nur so erhalten Sie mit den Script-Funktionen zur Fehlerbehandlung eine Fehlermeldung, wenn der Pfad nicht existiert. Beispiel: C:\AUTOMIC* Die Angabe eines oder mehrerer Volumes benötigt das Prefix "VOL=". Beispiel: "VOL=ALG*1" liefert Information zu allen Volumes, welche mit "ALG" beginnen, und mit "1" enden. Dazwischen dürfen beliebige Zeichen vorkommen. Hier sind es genau vier beliebige Zeichen, da die Bezeichnung eines Volumes 8-stellig ist. Bei der Bezeichnung von Generation Data Groups darf das Wildcardzeichen "*" nur innerhalb der Klammern verwendet werden.

Dateisystem-Wert	Die Script-Funktion kann folgende Informationen ermitteln: PATH_SPACE_ALLOCATED - Zugeteilter Speicher- bzw. Plattenplatz PATH_SPACE_RELEASE - Speicherplatz, der freigegeben werden kann (nur BS2000) PATH_SPACE_USED - Summe der Dateigrößen im angegebenen Pfad PATH_SPACE_UNUSED - Unbenutzter Speicher- bzw. Plattenplatz (nur BS2000) PATH_FILE_COUNT - Anzahl der Dateien, PATH_FOLDER_COUNT - Anzahl der Ordner (nur Windows und Unix), FILESYSTEM_SPACE_TOTAL - Speicherkapazität des Laufwerkes (nur Windows), FILESYSTEM_SPACE_USED - Belegter Speicher des Laufwerkes (nur Windows), FILESYSTEM_SPACE_FREE - Freier Speicherplatz von Volumes (z/OS) oder eines Laufwerkes (Windows) Format: AE-Name, Script-Literal oder Script-Variable
Maßeinheit	Optional kann angegeben werden, in welcher Form *Dateisystem-Wert* zurückgeben werden soll. Ohne *Maßeinheit* wird der Rückgabewert durch den Host bestimmt (Standard). Ein BS2000-Rechner gibt beispielsweise den Wert "1" für 1 PAM-Seite zurück. Das entspricht 2048 Bytes. Mit *Maßeinheit* wird der Rückgabewert wie angegeben umgerechnet. Erlaubte Werte: "Bytes", "KB", "MB", "GB" oder "TB". Format: AE-Name, Script-Literal oder Script-Variable Bei der Angabe einer ungültigen Maßeinheit wird ebenfalls der Standardwert verwendet. Die Script-Funktion bricht daher auch dann nicht ab, wenn Sie :ON_ERROR einsetzen.
Unterverzeichnisse einbeziehen	Beeinflusst, ob die Unterverzeichnisse des angegebenen Pfades durchsucht werden sollen. Erlaubte Werte: "Y" (Standardwert) und "N" Der Parameter ist nur für VMS-, UNIX- und Windows-Agenten wirksam. Beachten Sie, dass die Performance des Automation Engine Systems belastet wird, wenn Sie diese Option aktivieren!

Rückgabewerte
Ergebnis für den gesuchten Dateisystem-Wert "0" - Bei der Ermittlung des Dateisystem-Wertes ist ein Fehler aufgetreten (Ausnahme: PATH_FILE_COUNT siehe unten).

Anmerkungen

Dieses Sprachmittel kann nur in Zusammenhang mit Betriebssystem-Agenten (Windows, UNIX, VMS, z/OS, OS/400, NSK und BS2000) verwendet werden!

Tritt beim Zugriff auf Informationen zum Dateisystem ein Fehler auf (z.B: Pfad nicht gefunden), ist der Rückgabewert "0".

Mit der Script-Anweisung :ON_ERROR legen Sie die Reaktion auf diesen Fehler fest. Wie bisher können Sie ihn mit den Script-Funktionen zur Fehlerbehandlung analysieren. Das Script wird weiter prozessiert. Es besteht aber auch die Möglichkeit, die Prozessierung des Scripts abzubrechen.

⚠ Bei PATH_FILE_COUNT kann die Funktion natürlich auch 0 zurückliefern, wenn keine Dateien im Verzeichnis vorhanden sind. Verwenden Sie daher zusätzlich die Script-Funktionen zur Fehlerbehandlung (z.B. SYS_LAST_ERR_NR), um Fehler entdecken zu können. Beispielsweise, wenn der Host nicht aktiv ist. So ist es möglich zu unterscheiden, ob sich der Rückgabewert 0 auf die Dateianzahl oder auf einen aufgetretenen Fehler bezieht.

⚠ Bei fehlender Zugriffsberechtigung auf Systemordner (System Volume Information, Windows) wird ebenfalls Rückgabewert 0 ausgegeben.

ℹ Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

Optionale Parameter *Host* und *Pfad*

Fall 1:

GET_FILESYSTEM kann in den Script-Registerkarten von allen aktivierbaren Objekten (z.B. Workflow) verwendet werden. Beim Aufruf der Script-Funktion ist immer ein *Host* und ein *Pfad* anzugeben. Die Parameter können nur dann weggelassen werden, wenn GET_FILESYSTEM im Script schon einmal aufgerufen wurde. Sie erhalten aber in diesem Fall denselben Wert wie beim vorigen Aufruf. Es folgt dazu ein Beispiel:

```
:SET&AnzDateien# = GET_FILESYSTEM(WIN01, "C:\Temp", PATH_FILE_COUNT)
!diverse Scriptzeilen
:SET&AnzDateien# = GET_FILESYSTEM(,, PATH_FILE_COUNT)
!diverse Scriptzeilen
:SET&AnzDateien# = GET_FILESYSTEM(WIN01, "C:\Temp", PATH_FILE_COUNT)
```

Der erste Aufruf von GET_FILESYSTEM liefert die Dateianzahl im Verzeichnis C:\Temp (z.B. 50). Sollten jetzt Dateien in diesem Ordner gelöscht werden, gibt die zweite Verwendung der Script-Funktion auch weiterhin 50 aus. Der dritte Aufruf jedoch ermittelt die Dateisystem-Werte neu und liefert eine niedrigere Dateianzahl.

Fall 2:

Beim Eintreten eines Ereignisses vom Typ "Dateisystem", kann mit der Script-Funktion GET_FILESYSTEM auf eine Reihe Informationen zum Dateisystem und der Ausnutzung des Speicher- bzw. Plattenplatzes zugegriffen werden. Vom Agenten werden dabei alle Informationen übermittelt, die dann einzeln unter Angabe von *Dateisystem-Wert* abgefragt werden können. Die Script-Funktion wird dabei ohne Angabe von *Host* und *Pfad* aufgerufen, da diese bereits in der Registerkarte **Dateisystem** des Ereignisses definiert sind.

OS/400: Besonderheiten bei Dateisystem-Werten

Geben Sie immer sowohl eine Bibliothek als auch eine Datei an, um gültige Werte zum Dateisystem zu erhalten.

z/OS: Besonderheiten bei Dateisystem-Werten

PATH_SPACE_ALLOCATED - Der zugewiesene Speicherplatz in z/OS kann nicht ermittelt werden. Die Funktion gibt den benützten Speicherplatz zurück.
PATH_SPACE_USED - Benutzter Speicherplatz.

PATH_SPACE_UNUSED - Ist immer Null, da dieser Wert die Differenz von PATH_SPACE_ ALLOCATED und PATH_SPACE_USED ist.

Bei der Angabe einer GDG-Bezeichnung kann die Script-Funktion die folgenden Dateisystem-Werte ermitteln: PATH_SPACE_USED und PATH_FILE_COUNT.

Besonderheiten bei Dateisystem-Werten BS2000 gegenüber anderen Betriebssystemen

Nur in BS2000 können sich zugeteilter und benutzter Speicherplatz unterscheiden. Dort werden beispielsweise für eine Datei 1000 PAM-Seiten reserviert. Der Inhalt der Datei beträgt aber nur 100 PAM-Seiten.

Bei anderen Betriebssystemen, wie UNIX, VMS und MPE gilt folgendes:

PATH_FILE_COUNT - Anzahl der Dateien
PATH_FOLDER_COUNT - Anzahl der Ordner
PATH_SPACE_USED - Summe der Dateigrößen im angegebenen Pfad
PATH_SPACE_TOTAL - ist mit PATH_SPACE_USED identisch

Beispiele

Das Beispiel nutzt die Script-Funktion GET_FILESYSTEM, um die Anzahl vorhandener Dateien zu ermitteln und eine entsprechende Meldung abzusetzen. Da die beiden ersten Parameter nicht angegeben wurden, handelt es sich um den Script-Auszug eines Ereignisses.

```
:SET&ANZAHL# = GET_FILESYSTEM(,,PATH_FILE_COUNT)
:SEND_MSG"MEIER","EDV","Es wurden &ANZAHL# Datei(en) zur Verarbeitung
bereitgestellt."
```

Im folgenden Beispiel wird die Script-Funktion GET_FILESYSTEMim Script eines Jobs verwendet. Es werden alle verfügbaren Informationen zum Laufwerk abgerufen und im Aktivierungsprotokoll ausgegeben.

```
:SET&E1# = GET_FILESYSTEM(WIN01,"E:\",FILESYSTEM_SPACE_TOTAL,MB)
:SET&E2# = GET_FILESYSTEM(,,FILESYSTEM_SPACE_USED,MB)
:SET&E3# = GET_FILESYSTEM(,,FILESYSTEM_SPACE_FREE,MB)
:PRINT"Speicherkapazität des Laufwerkes: &E1# MB"
:PRINT "Belegter Speicher: &E2# MB"
:PRINT "Freier Speicher: &E3# MB"
```

Die folgende Beispiele zeigen die Verwendung der Script-Funktion mit GDG:

```
!Anzahl der Dateigenerationen der Gruppe TEST.XXX
:SET&DATEINAME# = GET_FILESYSTEM("MVSHOST", "TEST.XXX(*)", PATH_FILE_
COUNT)

!Summe des belegten Platzes der aktuellen Generation
:SET&PLATZ# = GET_FILESYSTEM("MVSHOST", "TEST.XXX(0)", PATH_SPACE_
USED)
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:ON_ERROR	Legt die Reaktion auf bestimmte Fehler bzw. Meldungen von Script-Sprachmitteln fest.

Script-Sprachmittel - Ereignisbehandlung
Beispielsammlung:
Darstellungen mittels Cockpit

Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.10.3 GET_WIN_EVENT

Script-Funktion: Ermittelt Einträge in das System-, Sicherheits- und Anwendungsprotokoll von Windows beim Eintreten eines Ereignisses.

Syntax

GET_WIN_EVENT*(Schlüsselwort*[, *Index*]*)*

Syntaxteil	Beschreibung/Format
Schlüsselwort	Name des Feldes, dessen Inhalt für diesen Ereigniseintrag ermittelt werden soll. Format: AE-name oder Script-Variable CATEGORY - Kategorie des Ereignisses EVENT_ID - Ereignis-ID INSERT - Greift auf einen bestimmten variablen Meldungsteil zu. INSERT_COUNT - Ermittelt die Anzahl der variablen Meldungsteile. SOURCE - Quelle des Ereignisses TIMESTAMP - Datum und Uhrzeit TYPE - Art des Ereignisses USER - Benutzer
Index	Zugriff auf die variablen Meldungsteile im Feld "Beschreibung" des Ereignisdetails. Format: Script-Literal, Zahl oder Script-Variable. Standard: "1" Kann nur in Verbindung mit dem Schlüsselwort INSERT verwendet werden (INSERT, *Index*).

Anmerkungen

Die Script-Funktion wird in einem Ereignis vom Typ "Konsole" für Windows verwendet. Mit diesem Ereignistyp kann die Ereignisanzeige von Windows überwacht werden. Wird ein Eintrag im System-, Sicherheits- oder Anwendungsprotokoll erkannt, welcher den Festlegungen in der Registerkarte "Detail" entspricht, tritt dieses Ereignis ein. Dabei werden die Verarbeitungsanweisungen aus der Registerkarte **! Script** prozessiert. Mit GET_WIN_EVENT kann nun mit einem *Schlüsselwort* auf bestimmte Informationen dieses Eintrages zugegriffen werden.

Wird als *Schlüsselwort* TYPE verwendet, gibt es folgende Rückgabewerte der Script-Funktion: "I" für Information, "W" für Warnung, "E" für Fehler, "S" für Überwachung-Erfolgreich oder "F" für Überwachung-Fehlversuch.

Meldungstexte bestehen in Microsoft Windows aus statischen und variablen Teilen. Die Script-Funktion liefert jedoch nur die variablen Meldungsteile. Mit INSERT_COUNT als *Schlüsselwort* kann zunächst die Anzahl der variablen Meldungsteile festgestellt werden. Mit INSERT als *Schlüsselwort* und der Angabe von *Index* wird auf einen bestimmten variablen Meldungsteil

zugegriffen. *Index* kann ausschließlich zu INSERT optional angegeben werden. Ohne *Index* gibt die Funktion den ersten variablen Meldungsteil zurück.

Beispiele

Im Beispiel wird als erstes die Anzahl der variablen Meldungsteile im Feld "Beschreibung" des Ereignisdetails ermittelt. Die Anzahl wird in der Script-Variablen "&COUNT" gespeichert. Danach wird eine Prozess-Schleife durchlaufen, in welcher der zweite und dritte variable Meldungsteil (Schlüsselwort INSERT) in den Aktivierungsreport geschrieben werden.

```
:SET &COUNT# = GET_WIN_EVENT("INSERT_COUNT")
:SET &IDX# = 1
:WHILE &IDX# <= &COUNT#
:   SET &INSERT# = GET_WIN_EVENT("INSERT", &IDX#)
:   SET &HELP# = FORMAT(&IDX#, "000")
:   PRINT "INSERT[&HELP#] = '&INSERT#'"
:   SET &IDX# = ADD(&IDX#, 1)
:ENDWHILE
```

Die gesamte Meldung des Ereignisdetails lautet beispielsweise: Der Benutzer "00432233778822#0001" hat eine Verbindung mit "T-Online" hergestellt, unter Verwendung des Gerätes "AVMISDN1". Die variablen Meldungsteile werden mit folgenden Zeilen im Aktivierungsreport protokolliert:

```
20010117/193135.000 - U0020408 INSERT[002] = 'T-Online'
20010117/193135.000 - U0020408 INSERT[003] = 'AVMISDN1'
```

Siehe auch:

Script-Sprachmittel - Ereignisbehandlung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.11 Systemzustände und -bedienung

3.11.1 :DISCONNECT

Script-Anweisung: Trennt eine Verbindung zum Automation Engine System.

Syntax

:DISCONNECT *Verbindung, Name*

Syntaxteil	Beschreibung/Format

Verbindung	Art der Verbindung. Format: Script-Literal, Script-Variable oder Zahl Erlaubte Werte: "USER", "HOST" und "HOSTGROUP" "USER" = Verbindung eines Benutzers oder einer BenutzerGruppe. "HOST" = Verbindung eines Agenten. "HOSTGROUP" = Verbindung von Agenten einer AgentenGruppe.
Name	Name eines Benutzers, einer BenutzerGruppe, eines Agenten oder einer AgentenGruppe. Format: Script-Literal, Script-Variable oder Zahl

Anmerkungen

Mit der Script-Anweisung können Verbindungen zum Automation Engine System getrennt werden.

Benutzer oder BenutzerGruppe müssen dem gleichen Mandanten angehören wie das Objekt, in dessen Script diese Script-Anweisung verwendet wird. Der Benutzer wird durch eine Meldung im UserInterface darauf aufmerksam gemacht, dass die Verbindung zur Automation Engine zwischenzeitlich unterbrochen war und er sich neu anmelden müsse. Bestanden keine aktiven Verbindungen der Benutzer zum Automation Engine System, führt dies nicht zu einem Fehler. Sind hingegen Benutzer oder BenutzerGruppe ungültig, wird die Script-Verarbeitung mit einer Fehlermeldung abgebrochen.

:DISCONNECT in Verwendung mit "HOST" oder "HOSTGROUP" beendet einen Agenten nicht, sondern trennt nur temporär die Verbindung. Der Agent verbindet sich wieder zur Automation Engine, sobald das nächste Signal zur Überprüfung der Agenten, gesendet wird. Das Trennen der Verbindung zu einem Agenten ist beispielsweise dann sinnvoll, wenn ein neuer Kommunikationsprozess gestartet wird und die Agenten sich umverteilen sollen. Verwenden Sie zum Beenden eines Agenten das Script-Sprachmittel :TERMINATE.

Anstatt eines einzelnen Agenten kann auch eine AgentenGruppe angegeben werden. Die Trennung der Verbindung wird für alle Agenten, die dieser Gruppe angehören, durchgeführt.

Um dieses Script-Anweisung ausführen zu können, benötigt ein Benutzer das Recht "Abbrechen" für den Benutzer bzw. Agenten.

Beispiel

Im Beispiel wird der Benutzer "MEIER/UC4" vom Automation Engine System getrennt.

```
:DISCONNECT "USER", "MEIER/UC4"
```

Hier wird der Agent WIN01 vom Automation Engine System getrennt.

```
:DISCONNECT "HOST", "WIN01"
```

Alle Datenbank-Agenten sind im folgenden Beispiel von der Trennung betroffen.

```
:DISCONNECT "HOSTGROUP", "HOSTG_DB_UNIX"
```

Siehe auch:

Script-Sprachmittel	Beschreibung

:TERMINATE	Beendet einen Agenten, einen Arbeits- oder Kommunikationsprozess.
:SHUTDOWN	Beendet ein Automation Engine System.

Script-Sprachmittel - Systemzustände und -bedienung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.11.2 :SET_UC_SETTING

Script-Anweisung: Ändert Systemeinstellungen im laufenden Betrieb.

Syntax

:SET_UC_SETTING *Einstellung, Komponente, Wert*

Syntaxteil	Beschreibung/Format
Einstellung	Systemeinstellung, welche geändert werden soll. Format: AE-name, Script-Literal oder Script-Variable Erlaubte Werte: "WORKLOAD_MAX", "WORKLOAD_MAX_FT", "WORKLOAD_MAX_JOB", "SET_TRACE", "SERVER_MODE" "WORKLOAD_MAX" = Maximale Anzahl an Ressourcen, die der Agent für FileTransfers und Jobs bereitstellt "WORKLOAD_MAX_FT" = Maximale Anzahl an Ressourcen, die der Agent für FileTransfers bereitstellt "WORKLOAD_MAX_JOB" = Maximale Anzahl an Ressourcen, die der Agent für Jobs bereitstellt "SET_TRACE" = Trace-Optionen für die Arbeitsprozesse eines Automation Engine Systems oder eines Agenten. "SERVER_MODE" = Typ des Serverprozesses.
Komponente	Komponente, für welche die Systemeinstellung geändert werden soll. Format: Script-Literal, Script-Variable, AE-Name oder Script-Funktion Für "WORKLOAD_MAX", "WORKLOAD_MAX_FT" und "WORKLOAD_MAX_JOB": Name eines aktiven Agenten oder einer AgentenGruppe. Für "SET_TRACE": Name eines Automation Engine Systems oder eines Agenten Für "SERVER_MODE": Name des Serverprozesses.

Wert	Neue Zuweisung für die Systemeinstellung. Format: Script-Literal, Script-Variable oder Script-Funktion Für "WORKLOAD_MAX", "WORKLOAD_MAX_FT" und "WORKLOAD_MAX_JOB": Wert zwischen "-1" und "100000" oder "UNLIMITED". ⚠ Beachten Sie, dass größere Werte als 100000 als "UNLIMITED" interpretiert werden! Für "SET_TRACE": Trace-Optionen, 16-stellig. Für "SERVER_MODE": Servertyp. Erlaubte Werte sind "D" für die Umschaltung zum Dialogprozess und "W" für die Umschaltung zum Arbeitsprozess. ⚠ Beachten Sie, dass ein Arbeitsprozess nicht zu einem Dialogprozess umgeschalten werden kann, wenn er eine Serverrolle ausübt!

Anmerkungen

Mit der Script-Anweisung :SET_UC_SETTING können derzeit drei Systemeinstellungen geändert werden. Die Änderungen sind solange gültig, bis ein neuer Wert zugewiesen wird bzw. die Serverprozesse oder der Agent beendet werden.

Die Berechtigung "Modifizieren zur Laufzeit" und das Privileg "Diagnoseinformationen erstellen" sind dafür erforderlich.

Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

Ändern von Trace-Optionen

Durch das Setzen der Trace-Optionen wird es möglich, aus einem Script heraus in einem Ausnahmefall das Verhalten der Arbeitsprozesse und Agenten systemnah zu protokollieren. Für den Trace müssen jene nicht beendet werden. Da aber unter Umständen umfangreiche Daten anfallen, kann es zu Performance-Verlusten kommen. Das Setzen von Trace-Optionen sollte nur in enger Abstimmung mit Support erfolgen.

Um die Trace-Optionen zu ändern, ist als *Einstellung* SET_TRACE anzugeben. *Wert* ist bei den Trace-Optionen ein String mit 16 Stellen. Jede Stelle entspricht einem bestimmten Traceflag, beispielsweise die erste Stelle einem TCP/IP-Trace. Die Reihenfolge der Traceflags ist identisch mit der Reihenfolge, die im Eigenschaften-Dialog in der Systemübersicht für Serverprozesse und Agenten verwendet wird.

Beachten Sie, dass bei einem Neustart von nur einem Arbeitsprozess, dieser die Trace-Optionen verwendet, die in der INI-Datei festgelegt wurden. Alle anderen benutzen den mit dieser Script-Anweisung eingestellten Wert, bis auch sie neu gestartet werden.

Beispiele

Im ersten Beispiel wird die Anzahl an Ressourcen, die der Agent WIN01 für Jobs und FileTransfers zur Verfügung stellt, auf 1000 gesetzt. Das Ergebnis wird zur Kontrolle im Aktivierungsprotokoll ausgegeben.

```
:SET_UC_SETTING WORKLOAD_MAX, WIN01, 1000
:SET &RET# = GET_UC_SETTING(WORKLOAD_MAX_JOB, WIN01)
:PRINT &RET#
```

Im zweiten Beispiel wird zuerst der Name des Automation Engine Systems ermittelt und dann der TCP/IP-Trace für dessen Arbeitsprozesse eingeschaltet.

```
:SET &TRC# = GET_UC_SYSTEM_NAME()
:SET_UC_SETTING SET_TRACE, &TRC#, "1000000000000000"
```

Im folgenden Beispiel wird der Serverprozess "UC4#WP003" zu einem Dialogprozess gemacht, wenn dieser aktiv ist.

```
:IF SYS_SERVER_ALIVE("UC4#WP003") = "Y"
:   SET_UC_SETTING "SERVER_MODE", "UC4#WP003", "D"
:ELSE
:   SEND_MSG "ADMIN","UC4","Arbeitsprozess UC4#WP003 ist nicht aktiv!"
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
GET_UC_SETTING	Liest aktuelle Systemeinstellungen.

Script-Sprachmittel - Systemzustände und -bedienung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.11.3 :SHUTDOWN

Script-Anweisung: Beendet ein Automation Engine System.

Syntax

:SHUTDOWN UC4

Syntaxteil	Beschreibung/Format
UC4	Konstante, die zum Beenden des Automation Engine Systems angegeben werden muss. Format: AE-name, Script-Literal oder Script-Variable

Anmerkungen

Mit der Script-Anweisung werden alle Arbeits- und Kommunikationsprozesse des Automation Engine Systems beendet.

Aus Sicherheitsgründen, um beispielsweise Verwechslungen mit JCL zu vermeiden, muss die Script-Anweisung immer zusammen mit der Konstanten **UC4** verwendet werden.

Für das Beenden einzelner Arbeits- und Kommunikationsprozesse oder auch Agenten wird :TERMINATE verwendet.

Um dieses Script-Anweisung ausführen zu können, benötigt ein Benutzer das Recht "Abbrechen" für den Serverprozess.

Beispiel

Im Beispiel wird das Automation Engine System beendet, wenn die aktuelle Tageszeit später als 20 Uhr ist.

```
:IF  SYS_TIME() > "200000"
:   SHUTDOWN UC4
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:DISCONNECT	Trennt eine Verbindung zum Automation Engine System.
:TERMINATE	Beendet einen Agenten, einen Arbeits- oder Kommunikationsprozess.

Script-Sprachmittel - Systemzustände und -bedienung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.11.4 :TERMINATE

Script-Anweisung: Beendet einen Agenten, die Agenten einer AgentenGruppe, einen Arbeits- oder Kommunikationsprozess.

Syntax

:TERMINATE [*Verbindung,*] ***Name***

Syntaxteil	Beschreibung/Format
Verbindung	Art der Verbindung. Format: AE-name, Script-Literal oder Script-Variable Erlaubte Werte: "HOST" (Standardwert), "HOSTGROUP" und "SERVER" "HOST" = Agent "HOSTGROUP" = Agenten einer AgentenGruppe "SERVER" = Arbeits- bzw. Kommunikationsprozess.
Name	Name eines Agenten, einer AgentenGruppe, eines Arbeits- oder Kommunikationsprozesses. Format: Script-Literal oder Script-Variable

Anmerkungen

Mit der Script-Anweisung können Agenten sowie Arbeits- und Kommunikationsprozesse beendet werden. Der Parameter *Verbindung* muss nicht angegeben werden, wenn das Ende eines Agenten eingeleitet werden soll. Es ist jedoch das Komma für den nicht verwendeten Parameter zu setzen.

Anstatt eines einzelnen Agenten kann auch eine AgentenGruppe angegeben werden. Die Beendigung der Verbindung wird für alle Agenten, die dieser Gruppe angehören, durchgeführt.

Soll die Verbindung zu einem Agenten nur kurzzeitig getrennt werden, empfiehlt es sich das Script-Sprachmittel :DISCONNECT zu verwenden. Hier verbindet sich der Agent wieder zum Server, sobald das nächste Signal zur Überprüfung der Agenten gesendet wird. Mit :SHUTDOWN können alle Arbeits- und Kommunikationsprozesse des Automation Engine Systems beendet werden.

Um dieses Script-Anweisung ausführen zu können, benötigt ein Benutzer das Recht "Abbrechen" für den Serverprozess bzw. Agenten.

Beispiele

Im Beispiel wird ein Arbeitsprozess beendet.

```
:TERMINATE SERVER,"UC4#WP001"
```

In diesem Beispiel wird der Agent "WIN21" beendet. Auf die Angabe der Verbindungsart wird dabei verzichtet.

```
:TERMINATE ,"WIN21"
```

Siehe auch:

Script-Sprachmittel/Script-Funktion	Beschreibung
:DISCONNECT	Trennt eine Verbindung zum Automation Engine System.
:SHUTDOWN	Beendet ein Automation Engine System.
MODIFY_SYSTEM	Führt ServiceManager-Aktionen oder Queue-Modifikationen durch.

Script-Sprachmittel - Systemzustände und -bedienung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.11.5 CHANGE_LOGGING

Script-Funktion: Veranlasst einen Wechsel der Log-Datei.

Syntax

CHANGE_LOGGING *(Komponente)*

Syntaxteil	Beschreibung/Format

Komponente	Name der Komponente dessen Log-Datei gewechselt werden soll. Format: AE-name oder Script-Variable Sie können folgende Komponenten angeben: • Kommunikationsprozess • Arbeitsprozess • Agent • AgentenGruppe

Rückgabewert
"0" - Der Wechsel der Log-Datei war erfolgreich. "20223" - Die angegebene AgentenGruppe existiert nicht. "20722" - Der angegebene Serverprozess bzw. Agent oder die AgentenGruppe existiert nicht bzw. ist nicht aktiv.

Anmerkungen

Mit dieser Script-Funktion kann ein Wechsel der Log-Datei erzwungen werden.

In der Variablen UC_SYSTEM_SETTINGS kann der Administrator systemweit die Anzahl der Tage bzw. die Größe in Megabyte festlegen, nach deren Erreichung die Log-Dateien gewechselt werden sollen. Ein temporärer Wechsel der Log-Datei ist in der Registerkarte "Eigenschaften" der Systemübersicht möglich. Die Änderungen gelten in diesem Fall bis zum nächsten Neustart. Der Speicherort für die Log-Dateien wird in der INI-Datei des Servers bzw. des Agenten festgelegt.

Bei der Angabe einer AgentenGruppe wird die Log-Datei bei allen zugehörigen Agenten gewechselt. Sie erhalten keine Fehlernummer als Rückgabewert, wenn einer der Agenten nicht aktiv ist und ein Wechsel der Log-Datei daher nicht vorgenommen werden kann.

⚠ Beachten Sie, dass ein Logwechsel eines Arbeitsprozesses automatisch für alle Arbeitsprozesse gilt.

Beispiel

Im Beispiel wird ein Logging-Wechsel für alle Arbeitsprozesse veranlasst.

```
:SET  &RET# = CHANGE_LOGGING ("UC4#WP001")
```

Siehe auch:

Script-Sprachmittel - Systemzustände und -bedienung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.11.6 GET_UC_SERVER_NAME

Script-Funktion: Ermittelt den Namen des Arbeitsprozesses, in dem das Script läuft.

Syntax

GET_UC_SERVER_NAME *()*

Rückgabewert
Name des Arbeitsprozesses.

Beispiel

Im Beispiel wird der Namen des Arbeitsprozesses (z.B. "UC4#WP001") ermittelt und als Ergebnis im Aktivierungsprotokoll ausgegeben.

```
:SET  &RET# = GET_UC_SERVER_NAME()
 :PRINT  &RET#
```

Siehe auch:

Script-Sprachmittel - Systemzustände und -bedienung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.11.7 GET_UC_SETTING

Script-Funktion: Liest aktuelle Systemeinstellungen.

Syntax

GET_UC_SETTING*(Einstellung,Komponente, Option)*

Syntaxteil	Beschreibung/Format

Einstellung	Systemeinstellung, welche gelesen werden soll. Format: AE-name, Script-Literal, Script-Variable oder Erlaubte Werte: "WORKLOAD_ACTUAL_FT", "WORKLOAD_ACTUAL_JOB", "WORKLOAD_MAX_FT", "WORKLOAD_MAX_JOB", "SET_TRACE", "SERVER_MODE", "SERVER_OPTIONS", "QUEUE" "WORKLOAD_ACTUAL_FT" = Gerade belegte Anzahl der Ressourcen, die der Agent für FileTransfers bereitstellt "WORKLOAD_ACTUAL_JOB" = Gerade belegte Anzahl der Ressourcen, die der Agent für Jobs bereitstellt "WORKLOAD_MAX_FT" = Maximale Anzahl an Ressourcen, die der Agent für FileTransfers bereitstellt "WORKLOAD_MAX_JOB" = Maximale Anzahl an Ressourcen, die der Agent für Jobs bereitstellt "SET_TRACE" = Trace-Optionen für die Arbeitsprozesse eines Automation Engine Systems "SERVER_MODE" = Typ des Serverprozesses "SERVER_OPTIONS" = Einstellungen am Server "QUEUE" = Einstellung / Wert eines Queue-Objektes
Komponente	Komponente, deren Systemeinstellung gelesen werden soll, abhängig vom Parameter *Einstellung*. Format: AE-Name, Script-Literal oder Script-Variable Für "WORKLOAD_ACTUAL_FT", "WORKLOAD_ACTUAL_JOB", "WORKLOAD_MAX_FT" und "WORKLOAD_MAX_JOB": Name des Agenten. Für "SET_TRACE": Name des Automation Engine Systems Für "SERVER_MODE": Name des Serverprozesses. Für "SERVER_OPTIONS" wird keine Komponente angegeben. Für "QUEUE": Name des Queue-Objektes.
Option	Option / Zustand des angegebenen Queue-Objektes, deren Wert gelesen werden soll. Format: AE-Name, Script-Literal oder Script-Variable Dieser Parameter ist nur anzugeben, wenn der Wert eines Queue-Objektes gelesen werden soll (*Einstellung* = QUEUE). Erlaubte Werte: "ACTIVE_COUNT", "CONSIDER_ERT", "MAX_SLOTS", "PRIORITY", "STATE" "ACTIVE_COUNT" = Anzahl der aktuell belegten Queue-Slots "CONSIDER_ERT" = Berücksichtigung der ERT für Ausnahmen "MAX_SLOTS" = Queue-Slot Maximum "PRIORITY" = Aktuelle Priorität (bei Ausnahmen kann der Wert von der Standard-Priorität abweichen) "STATE" = Aktueller Queue-Status

Rückgabewerte

Bei "WORKLOAD_ACTUAL_FT" und "WORKLOAD_ACTUAL_JOB":
Gerade belegte Anzahl der Ressourcen, die der Agent für FileTransfers bzw. Jobs bereitstellt.
"UNKNOWN" - Für den Agenten gibt es keine Beschränkung der Ressourcen.

Bei "WORKLOAD_MAX_FT" und "WORKLOAD_MAX_JOB":
Maximale Anzahl an Ressourcen, die der Agent für FileTransfers bzw. Jobs bereitstellt
"UNLIMITED" - Die Ressourcen sind unbegrenzt.

Bei "SET_TRACE":
Trace-Optionen des Arbeitsprozesses.
"0" - Es wurden keine Trace-Optionen gesetzt.

Bei "SERVER_MODE":
"C" - Kommunikationsprozess (CP)
"P" - Primärer Arbeitsprozess (PWP)
"W" - Arbeitsprozess (WP)
"D" - Dialogprozess (DWP)
"N" - NonStopprozess (NWP)
" " - Der Serverprozess ist nicht aktiv.

Bei "SERVER_OPTIONS":
Zeichenkette, welche die Server-Optionen enthält.

Bei "QUEUE" - "ACTIVE_COUNT":
Anzahl der verwendeten Queue-Slots

Bei "QUEUE" - "CONSIDER_ERT":
"1" - ERT wird beim Start von Aufgaben im Bezug auf Queue-Ausnahmen berücksichtigt
"2" - ERT wird nicht berücksichtigt

Bei "QUEUE" - "MAX_SLOTS":
Anzahl der maximalen Queue-Slots
"UNLIMITED" - Queue-Slots sind unbegrenzt

Bei "QUEUE" - "PRIORITY":
Aktuell Queue-Priorität

Bei "QUEUE" - "STATE":
"0" = GO
"1" = STOP

Anmerkungen

Die Trace-Optionen können in der Kategorie "Server" der Systemübersicht eingestellt werden. Hierfür stehen 16 Bereiche (z.B. TCP/IP) zur Verfügung. Die Script-Funktion liefert eine 16-stellige Zahl als Rückgabewert, wobei jede Nummer einem dieser Bereiche entspricht.

Die Server-Optionen werden vom Administrator in der Variablen UC_SYSTEM_SETTINGS mit dem Key SERVER_OPTIONS definiert. Die Script-Funktion liefert Ihnen die gesamte Zeichenkette. Verwenden Sie das Script-Sprachmittel MID, SUBSTR oder STR_CUT, um eine spezielle Server-Option auszulesen.

Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

Beispiel

Im Beispiel wird die maximale Anzahl der Ressourcen, die der Agent WIN01 für Jobs bereitstellt, ermittelt. Das Ergebnis wird im Aktivierungsprotokoll ausgegeben.

```
:SET &RET# = GET_UC_SETTING(WORKLOAD_MAX_JOB, "WIN01")
:PRINT &RET#
```

Im zweiten Beispiel werden die Trace-Optionen der Arbeitsprozesse des Automation Engine Systems namens "DEVPROD" ausgelesen.

```
:SET &RET# = GET_UC_SETTING(SET_TRACE, "DEVPROD")
```

Im folgenden Beispiel wird der Typ des Serverprozesses "UC4#WP003" ermittelt.

```
:SET &RET# = GET_UC_SETTING(SERVER_MODE, "UC4#WP003")
```

Die folgenden Scriptzeilen lesen aus den Server-Optionen die dritte Einstellung ab. Sie legt fest ob Statistiksätze beim Kaltstart geprüft werden sollen.

```
:SET &RET# = GET_UC_SETTING(SERVER_OPTIONS)
:SET &OPTION# = SUBSTR(&RET#,3,1)
```

Folgendes Beispiel zeigt die Ermittlung des aktuellen Status eines Queue-Objektes und gibt diesen im Aktivierungsprotokoll aus:

```
:SET &RET# = GET_UC_SETTING(QUEUE,QUEUE.JOBS,STATE)
:IF &RET# = 0
: PRINT "Queue QUEUE.JOBS ist aktiv"
:ELSE
: PRINT "Queue QUEUE.JOBS ist gestoppt"
:ENDIF
```

Im folgenden Beispiel wird das aktuelle Slot-Maximum aus einem Queue-Objekt gelesen und in den Aktivierunsreport geschrieben.

```
:SET &RET# = GET_UC_SETTING(QUEUE,QUEUE.JOBS,MAX_SLOTS)
: PRINT "Queue QUEUE.JOBS - Max Slots: &RET#"
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:SET_UC_SETTING	Ändert Systemeinstellungen im laufenden Betrieb.

Script-Sprachmittel - Systemzustände und -bedienung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.11.8 GET_UC_SYSTEM_NAME

Script-Funktion: Ermittelt den Namen des Automation Engine Systems.

Syntax

GET_UC_SYSTEM_NAME*()*

Rückgabewert
Name des Automation Engine Systems

Beispiel

Im Beispiel wird der Namen des Automation Engine Systems (z.B. "UC4PROD") ermittelt und als Ergebnis im Aktivierungsprotokoll ausgegeben.

```
:SET &RET# = GET_UC_SYSTEM_NAME()
:PRINT &RET#
```

Siehe auch:

Script-Sprachmittel - Systemzustände und -bedienung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.11.9 ILM

Script-Funktion: Steuert die ILM-Funktionalität.

Installation

[Installation] [Status] [Start und Stop] [Switch-Out] [Prüfung] [Löschen einer Partition]

Syntax

ILM (INSTALLED)

Syntaxteil	Beschreibung/Format
INSTALLED	Fragt ab, ob die Datenbank mit ILM partitioniert worden ist

Anmerkungen

Die Script-Funktion liefert die folgenden Rückgabewerte:

"Y" - Partitionierung mit ILM wurde installiert
"N" - Die Datenbank wurde nicht partitioniert

Beispiel

```
:SET &ILM# = ILM(INSTALLED)
```

Status

[Installation] [Status] [Start und Stop] [Switch-Out] [Prüfung] [Löschen einer Partition]

Syntax

ILM (ACTIVE)

Syntaxteil	Beschreibung/Format
ACTIVE	Fragt ab, ob ILM aktiv ist

Anmerkungen

Die Script-Funktion liefert die folgenden Rückgabewerte:

"Y" - ILM ist aktiv, d.h. neue Partitionen werden angelegt und es erfolgen Switch-Outs (letzteres betrifft nur den MS SQL Server).
"N" - ILM ist nicht aktiv, d.h. es erfolgt kein Partitionswechsel und keine Switch-Outs (letzteres betrifft nur den MS SQL Server).

Beispiel

```
:SET &ILM# = ILM(ACTIVE)
```

Start und Stop

[Installation] [Status] [Start und Stop] [Switch-Out] [Prüfung] [Löschen einer Partition]

Syntax

ILM (START)
ILM (STOP)

Syntaxteil	Beschreibung/Format
START	Aktiviert ILM
STOP	Deaktiviert ILM

Anmerkungen

Ist ILM aktiv, so werden neue Partitionen angelegt und es erfolgen Switch-Outs (letzteres betrifft nur den MS SQL Server).

Ist ILM nicht aktiv, so erfolgt kein Partitionswechsel und keine Switch-Outs (letzteres betrifft nur den MS SQL Server).

Die Script-Funktion liefert "0", wenn der Start bzw. der Stopp erfolgreich war oder die Nummer des Fehlers, der aufgetreten ist.

Beispiel

ILM wird deaktiviert.

```
:SET &ILM# = ILM(STOP)
```

Switch-Out

[Installation] [Status] [Start und Stop] [Switch-Out] [Prüfung] [Löschen einer Partition]

Syntax

ILM (SWITCHOUT *[, Prüfung]***)**

Syntaxteil	Beschreibung/Format
SWITCHOUT	Löst einen Switch-Out für die älteste Partition aus.
Prüfung	Einstellung, ob vor dem Switch-Out eine Prüfung erfolgen soll Erlaubte Werte: "CHECK" (Standardwert) und "NOCHECK" "CHECK" - Vor dem Switch-Out wird geprüft, ob die betroffene Partition Datensätze von Aufgaben enthält, die noch aktiv sind. Der Switch-Out entfällt, wenn das der Fall ist. "NOCHECK" - Der Switch-Out wird ohne vorherige Prüfung vorgenommen.

Anmerkungen

Switch-Outs gehören zum MS SQL Server. Dieser Funktionsaufruf ist daher für Oracle-Datenbanken nicht relevant.

⚠ Beachten Sie, dass die Script-Funktion zwar einen Switch-Out auslöst, aber nicht auf die Beendigung des Switch-Outs wartet!

Die Script-Funktion liefert "0", wenn der Switch-Out erfolgreich war oder die Nummer des Fehlers, der aufgetreten ist.

⚠ Beachten Sie, dass die Anzahl der Online-Partitionen, die der Administrator in der Variablen UC_ILM_SETTINGS mit dem Key ONLINE_PARTITIONS einstellt, nicht berücksichtigt wird!

Beispiel:

Es sind 4 Partitions online. Durch mehrmaliges Aufrufen der Script-Funktion können Sie erreichen, dass nur mehr 3, 2 oder 1 Partition online sind.

ℹ Der Switch-Out kann nicht für die gerade aktuelle Partition durchgeführt werden. Es muss immer zumindest eine Partition online sein.

ℹ Ein Switch-Out kann nur dann ohne Datenverlust vorgenommen werden, wenn die Partition keine Datensätze von Aufgaben enthält, die noch aktiv sind.

⚠ Beachten Sie, dass ein erzwungener Switch-Out trotz aktiver Aufgaben nur nach Absprache mit dem Automic Support vorgenommen werden sollte!

Beispiel

Vor dem Switchout erfolgt eine Prüfung, da der Standardwert für den zweiten Parameter "CHECK" ist.

```
:SET &ILM# = ILM(SWITCHOUT)
```

Prüfung

[Installation] [Status] [Start und Stop] [Switch-Out] [Prüfung] [Löschen einer Partition]

Syntax

ILM (CHECK*, Partitionsnummer*)

Syntaxteil	Beschreibung/Format
CHECK	Prüft, ob aktive Objekte in der angegebenen Partition vorhanden sind.
Partitionsnummer	Nummer der Partition Format: Script-Literal oder Script-Variable

Anmerkungen

Die Script-Funktion liefert "0", wenn keine aktive Objekte in der angegebenen Partition vorhanden sind.

Beispiel

Die Partition "25" wird geprüft.

```
:SET &ILM# = ILM(CHECK, "25")
```

Löschen einer Partition

[Installation] [Status] [Start und Stop] [Switch-Out] [Prüfung] [Löschen einer Partition]

Syntax

ILM (DROP*, Partition* *[, Prüfung]*)

Syntaxteil	Beschreibung/Format
DROP	Löscht die angegebene Partition

Partition	Name oder Nummer der Partition Format: Script-Literal oder Script-Variable Wenn Sie den Namen einer einzelnen Staging-Tabelle (MS SQL Server) angeben, erfolgt der Löschvorgang nicht mit den ILM-Datenbank-Benutzer. Stattdessen wird der Datenbank-Benutzer verwendet, den Sie in der INI-Datei der Automation Engine in der Sektion [ODBC] hinterlegt haben. Dieser muss über die erforderlichen Rechte verfügen!
Prüfung	Einstellung, ob vor dem Löschen eine Prüfung erfolgen soll (nur für Oracle-Datenbanken relevant) Erlaubte Werte: "CHECK" (Standardwert) und "NOCHECK" "CHECK" - Vor dem Löschen wird geprüft, ob die betroffene Partition Datensätze von Aufgaben enthält, die noch aktiv sind. Der Löschvorgang entfällt, wenn das der Fall ist. "NOCHECK" - Der Löschvorgang wird ohne vorherige Prüfung vorgenommen.

Anmerkung

Der Parameter *Prüfung* ist beim MS SQL Server nicht relevant, da dort nur Staging-Tabellen gelöscht werden können. Die Prüfung für diese Tabellen erfolgte bereits beim Switch-Out.

Ein Drop kann nur dann ohne Datenverlust vorgenommen werden, wenn die Partition keine Datensätze von Aufgaben enthält, die noch aktiv sind.

Die Script-Funktion liefert "0", wenn die angegebene Partition erfolgreich gelöscht werden konnte.

Beispiel

Die Partition "25" wird gelöscht.

```
:SET &ILM# = ILM(DROP, "25")
```

Siehe auch:

Script-Sprachmittel - Systemzustände und -bedienung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.11.10 MODIFY_SYSTEM

Script-Funktion: Führt ServiceManager-Aktionen oder Queue-Modifikationen durch.

Allgemeine Informationen

Das Scriptsprachmittel deckt 2 verschiedene Aufgaben ab:

1. Aktionen eines ServiceManagers durchführen, der mit dem Automation Engine System gekoppelt ist.
2. Werte von Queue-Objekte ändern oder deren Status wechseln (GO/STOP).

Die durchzuführende Aktion der Script-Funktion richtet sich nach den angegebenen Parametern. Siehe dazu die Syntaxbeschreibungen unten.

⚠ Beachten Sie, dass das Starten von Serverprozessen und Agenten mit dem Scriptsprachmittel nur dann möglich ist, wenn ein ServiceManager verfügbar ist, der richtig konfiguriert und mit dem Automation Engine System verbunden ist. Beachten Sie dazu die Einstellungen im Agenten bzw. Server-Objekt.

Rückgabewerte
"0" - Die definierte Aktion der Script-Funktion wurde erfolgreich durchgeführt. "11901" - Das angegebene Queue-Objekt konnte im Mandanten nicht gefunden werden. "20836" - Es wurde ein ungültiger Wert für den Parameter MODE angegeben. "20837" - Angabe eines ungültigen Wertes für die Priorität oder die max. Slots des Queue-Objektes. "11677" - ServiceManager-Aufruf konnte nicht durchgeführt werden: kein CP gestartet. "11678" - Server oder Agent wurden nicht gestartet. "11679" - Erfolgreiche Abfrage des ServiceManagers. "11680" - Es konnte keine Verbindung zum ServiceManager hergestellt oder der angegebene Prozess im ServiceManager nicht gefunden werden. "11681" - ServiceManager-Aufruf nicht möglich, da der angegebene Host-Typ nicht unterstützt wird. "11682" - ServiceManager-Aufruf nicht möglich: Der angegebene Agent / Serverprozess wurde nicht gefunden. "11683" - ServiceManager-Aufruf wird nicht durchgeführt, da der Server nicht mit dem angegebenen ServiceManager verknüpft ist oder die selbe Abfrage bereits durchgeführt wird.

Queue

[Queue] [ServiceManager]

Syntax

MODIFY_SYSTEM*(Aktion, Queue , Wert)*

Syntaxteil	Beschreibung/Format
Aktion	Legt fest, welche Änderung an Queue-Objekten durchgeführt werden soll. Format: Script-Literal oder Script-Variable Erlaubte Werte: "MODE", "MAX_SLOTS" oder "PRIORITY" "MODE" = Queue-Status, Start / Stopp der Ausführung von Queue-Aufgaben "MAX_SLOTS" = maximal parallel laufende Aufgaben (max. Slots) "PRIORITY" = Aufgaben-Priorität
Queue	Name des zu ändernden Queue-Objektes Format: Script-Literal oder Script-Variable

Wert	Wert für die zu ändernde Einstellung / Status (*Aktion*). Format: Script-Literal oder Script-Variable Erlaubte Werte, abhängig von der gewählten Änderung: "MODE": "GO" oder "STOP" "MAX_SLOTS": Zahl von "0" bis "99999" oder "UNLIMITED" "PRIORITY": Zahl von "0" bis "255"

Anmerkungen

Wird mit dem Script-Sprachmittel der Status von Queue-Objekten (Start/Stop) geändert (*Aktion*: MODE) ist der neue Status bis zur nächsten Modifikation gültig. Die Änderung der Priorität und der maximalen Queue-Slots bleibt solange erhalten, bis die Werte durch eineAusnahme oder manuell durch einen Benutzer geändert werden.

Werden die maximalen Slots von Queue-Objekten auf "UNLIMITED" geändert, gibt es keine Begrenzung für die parallel laufenden Aufgaben.

Beispiel

In dem folgenden Beispiel wird der Status des Queue-Objektes "QUEUE.JOBS" auf "GO" gesetzt:

```
:SET&RET# = MODIFY_SYSTEM("MODE", "QUEUE.JOBS", "GO")
:IF &RET# = "0"
: PRINT "Die Verarbeitung des Queue-Objektes QUEUE.JOBS wurde
erfolgreich aktiviert."
:ELSE
: PRINT "QUEUE.JOBS: Fehler beim Ändern des Status auf GO."
:ENDIF
```

ServiceManager

[Queue] [ServiceManager]

Syntax

MODIFY_SYSTEM*(Aktion, Name [, Server-Modus])*

Syntaxteil	Beschreibung/Format

Aktion	Legt die Aktion fest, die für Agenten, Serverprozesse oder das Automation Engine System durchgeführt werden soll. Format: Script-Literal oder Script-Variable Erlaubte Werte: "STARTUP", "TERMINATE", "CHANGE_MODE", "DISCONNECT" oder "SHUTDOWN" "STARTUP" = Start eines Agenten oder Serverprozesses "TERMINATE" = Agent oder Serverprozess beenden. "SHUTDOWN" = Komplettes Automation Engine System herunterfahren. "CHANGE_MODE" = Modus eines Server-Arbeitsprozesses (WP) ändern. "DISCONNECT" = Agenten-Verbindung trennen und neu aufbauen.
Name	Name des Agenten bzw. Serverprozesses, der gestartet, beendet oder dessen Modus geändert werden soll. Format: Script-Literal oder Script-Variable
Server-Modus	Modus auf den der angegebene WP geändert werden soll. Format: Script-Literal oder Script-Variable Dieser Parameter muss nur dann angegeben werden, wenn der Modus eines WPs geändert wird (*Aktion* = CHANGE_MODE). Erlaubte Werte: "D": Dialogprozess "W": Arbeitsprozess

Anmerkungen

Eine weitere Funktionalität des Scriptsprachmittels MODIFY_SYSTEM ist die Möglichkeit Serverprozesse und Agenten zu starten oder zu beenden bzw. den Modus von Server-Arbeitsprozessen zu ändern.

Die Aktion "STARTUP" ist nur möglich, wenn ein ServiceManager verfügbar ist und der zu startende Agent bzw. Serverprozess richtig konfiguriert wurde. Siehe dazu die Einstellungen in der Registerkarte **Attribute** des Agenten bzw Server-Objektes.

Um das Automation Engine System mit "SHUTDOWN" herunterzufahren, muss für den Parameter *Name* der Wert "UC4" verwendet werden (siehe: :SHUTDOWN).

Der Parameter *Server-Modus* muss angegeben werden, wenn der Modus eines Server-Arbeitsprozesses geändert werden soll (Aktion: "CHANGE_MODE"). Im Zusammenhang mit den Aktionen "STARTUP", "TERMINATE", "DISCONNECT" und "SHUTDOWN" ist die Verwendung des Parameters *Server-Modus* ungültig.

Beispiel

Beenden des Agenten "WIN01":

```
:SET &ACT# = MODIFY_SYSTEM("TERMINATE", "WIN01")
```

Siehe auch:

Script-Sprachmittel	Beschreibung

:SHUTDOWN	Beendet ein Automation Engine System.
SYS_HOST_ALIVE	Überprüft, ob ein bestimmter Host aktiv ist.
SYS_SERVER_ALIVE	Überprüft, ob ein bestimmter Serverprozess aktiv ist.
TOGGLE_SYSTEM_STATUS	Stoppt oder startet die automatische Verarbeitung eines gesamten Mandanten.

Script-Sprachmittel - Objekte lesen oder verändern
Beispielsammlung
Benachrichtigung mit variablem Meldungstext
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.11.11 SYS_BUSY_01

Script-Funktion: Liefert die prozentuale Auslastung des Automation Engine -Arbeitsprozesses, auf dem das Script ausgeführt wird , während der letzten Minute.

Syntax

SYS_BUSY_01*()*

Rückgabewert
Prozentuale Auslastung der Serverprozesse des Automation Engine-Systems.

Anmerkungen

Sie können die prozentuale Auslastung der Automation Engine auch mit der Script-Funktion SYS_INFO abfragen.

Beispiel

Die Auslastung der Serverprozesse wird abgefragt. Liegt sie über 80%, wird eine Meldung an einen Benutzer geschickt.

```
:IF SYS_BUSY_01() > 80
:   SEND_MSG BU,UC4,"Auslastung der Automation Engine über 80%"
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_BUSY_10	Liefert die prozentuale Auslastung des Automation Engine-Arbeitsprozesses, auf dem das Script ausgeführt wird , während der letzten 10 Minuten.
SYS_BUSY_60	Liefert die prozentuale Auslastung des Automation Engine-Arbeitsprozesses, auf dem das Script ausgeführt wird , während der letzten Stunde..

Script-Sprachmittel - Systemzustände und -bedienung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.11.12 SYS_BUSY_10

Script-Funktion: Liefert die prozentuale Auslastung des Automation Engine-Arbeitsprozesses, auf dem das Script ausgeführt wird , während der letzten 10 Minuten.

Syntax

SYS_BUSY_10*()*

Rückgabewert
Prozentuale Auslastung der Serverprozesse des Automation Engine-Systems.

Anmerkungen

Sie können die prozentuale Auslastung der Automation Engine auch mit der Script-Funktion SYS_INFO abfragen.

Beispiel

Die Auslastung der Serverprozesse wird abgefragt. Liegt sie über 80%, wird eine Meldung an einen Benutzer geschickt.

```
:IF SYS_BUSY_10() > 80
:   SEND_MSG BU,UC4,"Auslastung der Automation Engine über 80%"
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_BUSY_01	Liefert die prozentuale Auslastung des Automation Engine -Arbeitsprozesses, auf dem das Script ausgeführt wird , während der letzten Minute..
SYS_BUSY_60	Liefert die prozentuale Auslastung des Automation Engine-Arbeitsprozesses, auf dem das Script ausgeführt wird , während der letzten Stunde..

Script-Sprachmittel - Systemzustände und -bedienung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.11.13 SYS_BUSY_60

Script-Funktion: Liefert die prozentuale Auslastung des Automation Engine-Arbeitsprozesses, auf dem das Script ausgeführt wird , während der letzten Stunde..

Syntax

SYS_BUSY_60*()*

Rückgabewert
Prozentuale Auslastung der Serverprozesse des Automation Engine-Systems.

Anmerkungen

Sie können die prozentuale Auslastung der Serverprozesse auch mit der Script-Funktion SYS_INFO abfragen.

Beispiel

Die Auslastung der Automation Engine wird abgefragt. Liegt sie über 80%, wird eine Meldung an einen Benutzer geschickt.

```
:IF SYS_BUSY_60() > 80
:   SEND_MSG BU,UC4,"Auslastung der Automation Engine über 80%"
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_BUSY_01	Liefert die prozentuale Auslastung des Automation Engine -Arbeitsprozesses, auf dem das Script ausgeführt wird , während der letzten Minute.
SYS_BUSY_10	Liefert die prozentuale Auslastung des Automation Engine-Arbeitsprozesses, auf dem das Script ausgeführt wird , während der letzten 10 Minuten.

Script-Sprachmittel - Systemzustände und -bedienung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.11.14 SYS_HOST_ALIVE

Script-Funktion: Überprüft, ob ein bestimmter Host aktiv ist.

Syntax

SYS_HOST_ALIVE*(Host *[, Verbindung]**)***

Syntaxteil	Beschreibung/Format
Host	Name des Agenten dessen Aktivität überprüft werden soll. Format: AE-name oder Script-Variable
Verbindung	Name eines R3 oder DB Verbindungs-Objektes mit dem die Verfügbarkeit des SAP-Systemes bzw. der Datenbank überprüft werden soll. Format: AE-Name oder Script-Variable Geben Sie bei SAP ein Verbindungs-Objekt an, das Sie im Agent-Objekt in der Registerkarte **Agent** eingestellt haben! Wird ein DB Verbindungs-Objekt angegeben, ist der Parameter *Host* optional! ⚠ Dieser Parameter kann nur verwendet werden, wenn für *Host* entweder ein SAP-Agent oder ein Datenbank-Agent angegeben wird! Dabei wird nicht geprüft, ob der Datenbank-Agent für die Auflösung von SQL-Variablen gestartet wurde.

Rückgabewerte
Bei der Angabe eines Hosts: "Y" - Der Agent ist aktiv. "N" - Der Agent ist nicht aktiv. Bei der zusätzlichen Angabe eines Verbindungs-Objektes: "Y" - Der Agent ist aktiv und das SAP-System bzw. die Datenbank ist verfügbar. "N" - Der Agent ist aktiv. Er kann allerdings keine Verbindung zum SAP-System bzw. zur Datenbank aufbauen. "?" - Der Agent ist nicht aktiv. Es kann nicht ermittelt werden, ob das SAP-System bzw die Datenbank verfügbar ist.

Anmerkungen

Betriebssystem-Agenten

Die Script-Funktion prüft, ob der Agent aktiv ist. Der Parameter *Verbindung* kann nicht angegeben werden.

ERP-Agenten

Für die ERP-Agenten Peoplesoft, OracleApplications und Siebel wird auch dann der Wert "N" zurückgeliefert, wenn der Agent zwar aktiv, die Enterprise Business Solution jedoch nicht verfügbar ist.

Bei SAP-Agenten haben Sie die Wahl. Geben Sie nur den Parameter *Host* an, wenn Sie überprüfen möchten, ob der SAP-Agent aktiv ist. Benötigen Sie die Information, ob eine das SAP-System erreichbar ist, können Sie das zugehörige SAP Verbindungs-Objekt angeben.

⚠ Beachten Sie, dass der SAP-Agent manche Verbindungen erst bei Bedarf öffnet (HTTP) oder nach längerer Nichtbenutzung automatisch schließt (ABAP). Existiert beim Aufruf von SYS_HOST_ALIVE keine Verbindung zum SAP-System, so wird eine neue Verbindung aufgebaut, damit geprüft werden kann, ob das SAP-System erreichbar ist. Beachten Sie in diesem Zusammenhang folgende wichtige Hinweise:

- Ist das SAP-System nicht erreichbar, müssen Sie möglicherweise auf ein langes Timeout warten.
- Besteht zwar eine Verbindung zum SAP-System muss trotzdem eine neue Verbindung aufgebaut werden, wenn diese gerade verwendet wird. Damit wird sichergestellt, ob das SAP-System tatsächlich noch reagiert.
- Das zusätzliche Aufbauen einer Verbindung kann auch zu Fehlern führen, wenn z.B: das CPIC-Limit bei SAP überschritten wird. In diesem Fall liefert SYS_HOST_ALIVE den Rückgabewert "N", obwohl über bereits bestehende Verbindungen Jobs ausgeführt werden.
- Die Änderung eines Verbindungs-Objektes wirkt sich erst nach einem Neustart des SAP-Agenten aus!

Datenbank-Agent

Wie bei SAP haben Sie die Möglichkeit nur den Agenten oder zusätzlich ein Verbindungs-Objekt (Typ: DB) anzugeben. Mit der Angabe eines Verbindungs-Objekt an, haben Sie die Möglichkeit zusätzlich die Verfügbarkeit der jeweiligen Datenbank zu überprüfen.

Beispiel

Im Beispiel wird geprüft, ob der Windows-Agent "WIN21" aktiv ist. Ist der Agent nicht aktiv, wird eine Meldung an den Administrator geschickt.

```
:IF SYS_HOST_ALIVE("WIN21") = "N"
:   SEND_MSG "ADMIN","automic","Agent WIN21 ist nicht aktiv!"
:ENDIF
```

Das zweite Beispiel ermittelt, ob der Agent SAP01 das SAP-System erreichen kann.

```
:SET &STATUS# = SYS_HOST_ALIVE("SAP01", "CONN.R3.ECC.ABAP")
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_ACT_HOST	Ermittelt den Namen des Hosts.

Script-Sprachmittel - Systemzustände und -bedienung
Beispielsammlung:
Darstellungen mittels Cockpit
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.11.15 SYS_INFO

Script-Funktion: Liest Informationen über das Automation Engine System aus.

Automation Engine-Version

[AE-Version] [Auslastung]

Syntax

SYS_INFO*(Komponente,* VERSION *[, [Typ] [, Komponenten-Name]])*

Syntaxteil	Beschreibung/Format
Komponente	Komponente deren Version ermittelt werden sollen Format: Script-Literal oder Script-Variable Erlaubte Werte: "SERVER", "AGENT" und "INITIALDATA" "SERVER" - UC4 Automation Engine "INITIALDATA" - Initialdaten der Datenbank "AGENT" - Agent. Der Name des Agenten ist mit dem Parameter *Komponenten-Name* anzugeben.
VERSION	Liefert die AE-Version der Komponente
Typ	Angabe, welcher Teil der Version ausgelesen werden soll. Format: AE-Name, Script-Variable oder Script-Literal Erlaubte Werte: "MAJOR" - Major-Version "MINOR" - Minor-Version "PATCH" - Service-Pack Nummer "RELEASE_IDENTIFIER" - Art des Builds und Buildnummer "ALL" (Standardwert) - Vollständige Version
Komponenten-Name	Name des Agenten, dessen Version ermittelt werden soll. Format: AE-Name, Script-Variable oder Script-Literal

Rückgabewerte
Entsprechende AE-Version der gewünschten Komponente "20291" - Der angegebene Agent wurde nicht gefunden "20680" - Unbekannter Wert für den Versions-Teil "20863" - Für die Komponente wurde ein ungültiger Wert angegeben "20864" - Der zweite Parameter lautet nicht VERSION.

Anmerkungen

Die Script-Funktion ermittelt die Version der Automation Engine, eines Agenten oder der Initialdaten der Datenbank. Mehr Informationen zum Aufbau der Versionsnummer finden Sie im Kapitel Bezeichnung der AE-Versionen .

Um die Version eines Agenten auszulesen, ist der Parameter *Komponente* auf den Wert "AGENT" zu setzen und bei *Komponenten-Name* der Name des Agenten anzugeben.

Die AE-Version setzt sich aus verschiedenen Informationen zusammen (Major-Version, Minor-Version, ...). Um nur einen bestimmten Teil der Versionen auszulesen, kann der Parameter *Typ* verwendet werden.

Beispiele

Im ersten Beispiel wird die Version der Automation Engine ausgelesen.

```
:SET &VERSION# = SYS_INFO(SERVER, VERSION)
```

Das zweite Beispiel ermittelt die Version der Initialdaten der Datenbank.

```
:SET &VERSION# = SYS_INFO(INITIALDATA, VERSION)
```

Im dritten Beispiel wird die Version des Agenten WIN01 ausgelesen:

```
:SET &VERSION# = SYS_INFO(AGENT, VERSION,,"WIN01")
```

Das vierte Beispiel ermittelt die einzelnen Teile der Version der Automation Engine (Major-, Minor-, Service-Pack - Version und Build-Nummer) und gibt diese im Aktivierungsreport aus. Im Anschluss wird auch die vollständige Versionsnummer ausgegeben.

```
:SET&VERSION# = SYS_INFO(SERVER, VERSION,MAJOR)
:PRINT "Automation Engine - Major Version: &VERSION#"
:SET&VERSION# = SYS_INFO(SERVER, VERSION,MINOR)
:PRINT "Automation Engine - Minor Version: &VERSION#"
:SET&VERSION# = SYS_INFO(SERVER, VERSION,PATCH)
:PRINT "Automation Engine - Service-Pack: &VERSION#"
:SET&VERSION# = SYS_INFO(SERVER, VERSION,RELEASE_IDENTIFIER)
:PRINT "Automation Engine - Build: &VERSION#"
:SET&VERSION# = SYS_INFO(SERVER, VERSION,ALL)
:PRINT "Automation Engine - Version: &VERSION#"
```

Beispiel für die Ausgabe im Aktivierungsprotokoll:

```
2013-05-07 16:16:52 - U0020408 Automation Engine - Major Version: 10
2013-05-07 16:16:52 - U0020408 Automation Engine - Minor Version: 0
2013-05-07 16:16:52 - U0020408 Automation Engine - Patch: 0
2013-05-07 16:16:52 - U0020408 Automation Engine - Build: -
dev+build.954
2013-05-07 16:16:52 - U0020408 Automation Engine - Version: 10.0.0-
dev+build.954
```

Auslastung

[AE-Version] [Auslastung]

Syntax

SYS_INFO*(*MQPWP*, BUSY*, Zeitraum*)
SYS_INFO*(Nachrichten-Warteschlange,* COUNT*)*
SYS_INFO*(Nachrichten-Warteschlange,* LENGTH*, Zeitraum)*

Syntaxteil	Beschreibung/Format

Nachrichten-Warteschlange	Nachrichten-Warteschlange zu der Informationen ermittelt werden sollen Format: Script-Literal oder Script-Variable Erlaubte Werte: "MQPWP", "MQWP" , "MQDWP", "MQOWP" und "MQRWP" "MQPWP" - Nachrichten-Warteschlange des primären Arbeitsprozesses "MQWP" - Nachrichten-Warteschlange der Arbeitsprozesse "MQDWP" - Nachrichten-Warteschlange der Dialogprozesse "MQOWP" - Nachrichten-Warteschlange für Outputs "MQRWP" - Nachrichten-Warteschlange für Ressource-Berechnungen
BUSY	Liefert die prozentuale Auslastung der Automation Engine.
COUNT	Liefert die Anzahl der Nachrichten in der Warteschlange.
LENGTH	Liefert die durchschnittliche Dauer für die Abarbeitung der Nachrichten-Warteschlange.
Zeitraum	Zeitspanne, die bei der Berechnung der Auslastung bzw. der durchschnittlichen Abarbeitungszeit herangezogen wird. Format: Script-Literal oder Script-Variable Erlaubte Werte: "01", "10" und "60" "01" - die letzte Minute "10" - die letzten 10 Minuten "60" - die letzte Stunde

Rückgabewerte
"20876" - Die Nachrichten-Warteschlange existiert nicht. "20864" - Der zweite Parameter ist ungültig. "20877" - Der Zeitraum entspricht nicht den erlaubten Werten. Bei BUSY: Prozentuale Auslastung der Automation Engine Bei COUNT: Anzahl der Nachrichten in der Warteschlange Bei LENGTH: Durchschnittliche Dauer für die Abarbeitung der Nachrichten-Warteschlange

Anmerkungen

Die Script-Funktion ermittelt Daten zu den Nachrichten-Warteschlangen.

Die prozentualle Auslastung der Automation Engine kann nicht nur mit dem Schlüsselwort BUSY abgefragt werden, sondern auch mit den Script-Funktionen SYS_BUSY_01, SYS_BUSY_10 und SYS_BUSY_60.

Beispiele

Das erste Beispiel fragt die Auslastung der Automation Engine während der letzten 10 Minuten ab. Liegt sie über 80%, wird eine Meldung an einen Benutzer geschickt.

```
:IF SYS_INFO(MQPWP, BUSY, "10") > 80
:   SEND_MSG MEIER,UC4,"Auslastung der Automation Engine über 80%"
:ENDIF
```

Das zweite Beispiel liest die Anzahl der Nachrichten aus, die sich in der Warteschlange der Dialogprozesse befinden.

```
:SET &ANZAHL# = SYS_INFO(MQDWP, COUNT)
```

Das dritte Beispiel liefert die aktuelle Abarbeitungszeit der Nachrichten-Warteschlange der Arbeitsprozesse während der letzten Stunde.

```
:SET &DAUER# = SYS_INFO(MQWP, LENGTH, "60")
```

Siehe auch:

Systemzustände und -bedienung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.11.16 SYS_SERVER_ALIVE

Script-Funktion: Überprüft, ob ein bestimmter Serverprozess aktiv ist.

Syntax

SYS_SERVER_ALIVE*(Serverprozess)*

Syntaxteil	Beschreibung/Format
Serverprozess	Name des Serverprozesses, dessen Aktivität überprüft werden soll. Format: Script-Literal oder Script-Variable

Rückgabewerte
"Y" - Der Serverprozess ist aktiv. "N" - Der Serverprozess ist nicht aktiv. "20349" - Der Serverprozess existiert nicht.

Anmerkungen

Wird der angegebene Serverprozess nicht gefunden, können Sie auf diesen Fehler mit :ON_ERROR reagieren und mit den Script-Funktionen zur Fehlerbehandlung analysieren. Das Script wird je nach Einstellung entweder weiter prozessiert oder abgebrochen.

Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

Beispiel

Im Beispiel wird geprüft, ob der Arbeitsprozess "UC4#WP005" aktiv ist. Ist der Arbeitsprozess nicht aktiv, wird eine Meldung an den Administrator geschickt.

```
:IF SYS_SERVER_ALIVE("UC4#WP005") = "N"
:   SEND_MSG "ADMIN","UC4","Arbeitsprozess UC4#WP005 ist nicht aktiv!"
:ENDIF
```

Siehe auch:

Script-Sprachmittel - Systemzustände und -bedienung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.11.17 SYS_SNMP_ACTIVE

Script-Funktion: Prüft, ob die SNMP-Kopplung (Simple Network Management Protocol) von AE aktiv ist.

Syntax

SYS_SNMP_ACTIVE*()*

Rückgabewerte
"Y" - Die SNMP-Kopplung ist in der AE aktiv. "N" - Die SNMP-Kopplung ist in der AE nicht aktiv.

Beispiel

Im Beispiel wird geprüft, ob die SNMP-Kopplung aktiv ist. Das Ergebnis wird im Aktivierungsprotokoll ausgegeben.

```
:SET  &RET# = SYS_SNMP_ACTIVE()
:PRINT  &RET#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:SEND_SNMP_TRAP	Sendet einen SNMP-Trap.

AE und SNMP
Script-Sprachmittel - Systemzustände und -bedienung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.11.18 SYS_USER_LANGUAGE

Script-Funktion: Liefert die Sprache in der der Server die Logdateien erzeugt.

Syntax

SYS_USER_LANGUAGE*()*

Rückgabewerte
"D" - Deutsch "E" - Englisch "F" - Französisch

Anmerkungen

Die Sprache in der die Server-Logdateien verfasst werden, können Sie in der INI-Datei ucsrv.ini mit dem Parameter language= festlegen. Diese Script-Funktion ermittelt das dort eingetragene Sprachkennzeichen.

Beispiel

Im Beispiel wird das mit der Script-Funktion ermittelte Sprachkennzeichen in das Aktivierungsprotokoll geschrieben.

```
:SET  &LOGLANG# = SYS_USER_LANGUAGE()
:PRINT  "Sprache des Servers ist &LOGLANG#."
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_USER_ALIVE	Prüft, ob der Benutzer mit einem UserInterface an AE angemeldet ist.
SYS_USER_DEP	Liefert die Abteilung des Benutzers, der die Aufgabe startete.
SYS_USER_LNAME	Liefert den Vor- und Nachnamen des Benutzers, der die Aufgabe startete.
SYS_USER_NAME	Liefert den Namen des Benutzers, der die Aufgabe startete.

Script-Sprachmittel - Benutzerdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.11.19 TOGGLE_SYSTEM_STATUS

Script-Funktion: Stoppt oder startet die automatische Verarbeitung eines gesamten Mandanten.

Syntax

TOGGLE_SYSTEM_STATUS*(Status)*

Syntaxteil	Beschreibung/Format
Status	Statusänderung Format: AE-name, Script-Literal oder Script-Variable Erlaubte Werte: "STOP" und "GO" "STOP" - Stoppt die automatische Verarbeitung eines Mandanten. "GO" - Startet die automatische Verarbeitung eines Mandanten.

Rückgabewerte
"0" - Der Mandantenstatus wurde erfolgreich geändert. "9" - Das erforderliche Privileg zum Ändern des Mandantenstatus fehlt.

Anmerkungen

Die Script-Funktion setzt den Systemstatus auf STOP oder auf GO.

Zur Ausführung der Script-Funktion ist das Privileg "Systemstatus ändern (STOP/GO)" notwendig!

Das Script-Sprachmittel bewirkt, dass die offenen Transaktionen des Scripts in die Datenbank geschrieben werden.

Beispiel

Im Beispiel wird die automatische Verarbeitung gestoppt. Der Rückgabewert der Funktion wird im Report ausgegeben.

```
:SET &RET# = TOGGLE_SYSTEM_STATUS(STOP)
:PRINT &RET#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
TOGGLE_OBJECT_STATUS	Stoppt oder startet die automatische Verarbeitung von AblaufPlänen, Gruppen, Ereignissen oder Schedules.

Script-Sprachmittel - Systemzustände und -bedienung
Systemstatus ändern
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12 Datum und Zeit

3.12.1 ADD_DAYS

Script-Funktion: Addiert Tage zu angegebenem Datum.

Syntax

ADD_DAYS*(Datum, Tage[, Kalender, Kalenderbegriff])*

Syntaxteil	Beschreibung/Format
Datum	Angabe eines Datums im Format "JJMMTT" oder "JJJJMMTT".Format: Script-Literal oder Script-VariableEs ist möglich, das Datum in einem anderen Datumsformat anzugeben. Geben Sie dazu zuerst das gewünschte Datumsformat, ein Trennzeichen (: oder ;) und danach das Datum an. Die Angabe des Datumsformats ist optional.
Tage	Anzahl von Tagen, mit denen gerechnet werden soll. Format: Script-Literal, Zahl ohne Hochkommata oder Script-Variable
Kalender	Name des Kalender-Objektes, der bei Berechnung zu berücksichtigen ist. Format: AE-name, Script-Literal oder Script-Variable
Kalenderbegriff	Name des Kalenderbegriffes, der bei Berechnung zu berücksichtigen ist. Format: AE-Name, Script-Literal oder Script-Variable

Rückgabewert
Datum im angegebenen Format "0" - Dem Kalenderbegriff wurden keine Tage zugewiesen oder das berechnete Datum liegt außerhalb des Kalenders. "20327" - Das Kalender-Objekt existiert nicht. "20328" - Der Kalenderbegriff existiert nicht im Kalender-Objekt.

Anmerkungen

Mit der Script-Funktion addieren Sie *Tage* zu einem angegebenen *Datum*. Optional können Sie diese Rechenoperation auch in Abhängigkeit von einem *Kalender* und einem *Kalenderbegriff* ausführen. Es werden dann nur solche *Tage* berücksichtigt, die im Kalenderbegriff definiert worden sind.

Wird ein bestimmtes Datumsformat angegeben, entspricht das zurückgegebene Datum diesem Format. Wird kein Datumsformat angeben, müssen die Formate "JJMMTT" oder "JJJJMMTT" für das Datum verwendet werden. Es wird im selben Format zurückgegeben. Als Trennzeichen zwischen Datumsformat und *Datum* sind Doppelpunkt oder Semikolon erlaubt.

Insbesondere sei auf die Möglichkeit hingewiesen, die Anzahl der *Tage* gleich Null zu setzen. Wenn das angegebene *Datum* in den *Kalenderbegriff* fällt, wird das *Datum* selbst zurückgegeben, sonst das folgende gültige *Datum* laut *Kalenderbegriff*.

⚠ Bei der Datumsberechnung wird der Bereich des angegebenen Kalenders berücksichtigt. Dieser hängt von der Einstellung ab, die der Administrator in der Variablen UC_CLIENT_SETTINGS mit den Keys NOW_MINUS und NOW_PLUS konfiguriert hat. Fällt das errechnete Datum außerhalb des Kalenders, liefert die Script-Funktion den Wert "0" zurück.

Beispiele

Das erste Beispiel addiert zwei Tage zu einem angegebenen Datum. Das Ergebnis ("000401") wird im Aktivierungsprotokoll ausgegeben.

```
:SET   &DATUM# = ADD_DAYS("000330", 2)
:PRINT   &DATUM#
```

In diesem Beispiel wird das Datum mit einer Script-Variablen übergeben. Da kein Datumsformat angegeben wird, verwendet jedes Datum das Defaultformat "JJMMTT". Das Ergebnis ("000401") wird auch hier im Aktivierungsprotokoll ausgegeben.

```
:SET   &DATUM1# = "000330"
:SET   &DATUM2# = ADD_DAYS(&DATUM1#, 2)
:PRINT   &DATUM2#
```

Im dritten Beispiel wird der nächste gültige Arbeitstag ermittelt. Dazu werden die Definitionen des Firmenkalenders herangezogen. Da das aktuelle Tagesdatum nicht im Defaultformat abgefragt wurde, muss auch bei der Script-Variablen das spezielle Datumsformat angegeben werden.

```
:SET   &AKTDAT# = SYS_DATE("TT.MM.JJ")
:SET   &NARBTAG# = ADD_DAYS("TT.MM.JJ:&AKTDAT#", 1, FIRMA, ARBEITSTAG)
```

Siehe auch:

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.2 ADD_PERIOD

Script-Funktion: Addiert eine Periode zu einem angegebenen Datum.

Syntax

ADD_PERIOD*(Datum, Periodenformat:Periode*[*, Ausgabeformat*]*)*

Syntaxteil	Beschreibung/Format

Datum	Angabe eines Datums im Format "JJMMTT" oder "JJJJMMTT". Format: Script-Literal oder Script-Variable Es ist möglich, das Datum in einem anderen Datumsformat anzugeben. Geben Sie dazu zuerst das gewünschte Datumsformat, ein Trennzeichen (: oder ;) und danach das Datum an. Die Angabe des Datumsformats ist optional.
Periodenformat:Periode	Format: Script-Literal oder Script-Variable Periode: Angabe einer Periode, die zum angegebenen Datum addiert werden soll. Periodenformat: Formatvorgabe für die *Periode*. Als Trennzeichen zwischen dem *Periodenformat* und der *Periode* ist ein Doppelpunkt (:) oder ein Semikolon (;) erlaubt.
Ausgabeformat	Formatvorgabe für das ermittelte Datum. Format: AE-name, Script-Literal oder Script-Variable Standardwert: "JJMMTT"

Rückgabewert
Datum im angegebenen Format.

Anmerkungen

Die Script-Funktion zählt zu einem angegebenen Datum Jahre, Monate, Quartale oder Wochen hinzu.

Der Script-Funktion wird ein Datum übergeben. Eine explizite Angabe des Datumsformats ist dabei optional. Wird kein bestimmtes Datumsformat verwendet, muss das Datum in den Formaten "JJMMTT" oder "JJJJMMTT" angegeben werden. Als Trennzeichen zwischen Datumsformat und *Datum* sind Doppelpunkt oder Semikolon erlaubt.

Zum angegebenen *Datum* wird die *Periode* addiert. *Periode* kann eine beliebige Zahl sein. Ist das Ergebnis ein Datum, welches nach dem 31.12.9999 liegen würde, führt das zu einem Fehler.

Ausgabeformat ist optional. Wird *Ausgabeformat* nicht verwendet, liefert die Script-Funktion ein Datum mit dem Defaultformat "JJMMTT" zurück.

Beispiele

Das erste Beispiel addiert zwei Wochen zum 6.3.2000. Das Ergebnis (20.03.2000) wird im Report ausgegeben.

```
:SET  &DATUM#=ADD_PERIOD ("TT.MM.JJ:06.03.00","WW;2","TT.MM.JJJJ")
:PRINT  &DATUM#
```

Das zweite Beispiel addiert ein Quartal zum 31.1.2000. Das Ergebnis (30-04-2000) wird im Report ausgegeben.

```
:SET  &DATUM#=ADD_PERIOD ("000131","Q:1","TT-MM-JJJJ")
:PRINT  &DATUM#
```

Im dritten Beispiel wird ein Jahr zum 29.2.2000 addiert. Als Ergebnis wird 28.02.2001 im Report ausgegeben, denn das Jahr 2001 ist kein Schaltjahr.

```
:SET  &DATUM#=ADD_PERIOD ("20000229","JJ:0001",TT.MM.JJJJ)
:PRINT  &DATUM#
```

Siehe auch:

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.3 ADD_TIME

Script-Funktion: Addiert zwei Zeiten.

Syntax

ADD_TIME*(Zeit1, Zeit2 *[,Ausgabeformat]**)***

Syntaxteil	Beschreibung/Format
Zeit1 und *Zeit2*	Angabe einer Zeit im Format "HHMMSS". Format: Script-Literal oder Script-Variable Es ist möglich, die Zeit in einem anderen Zeitformat anzugeben. Geben Sie dazu zuerst das gewünschte Zeitformat, ein Trennzeichen (;) und danach die Zeit an. Die Angabe des Zeitformats ist optional.
Ausgabeformat	Formatvorgabe für die ermittelte Zeit. Format: AE-name, Script-Literal oder Script-Variable Standard: "HHMMSS"

Rückgabewert
Zeit im angegebenen Format

Anmerkungen

Mit der Script-Funktion addieren Sie zwei Zeiten. Dabei wird der Wechsel von 23:59:59 auf 00:00:00 berücksichtigt.

Der Script-Funktion werden zwei Zeiten übergeben. Die Angabe eines bestimmten Zeitformats ist dabei optional. Wird kein Zeitformat verwendet, muss die Zeit im Format "HHMMSS" angegeben werden. Als Trennzeichen zwischen Zeitformat und Zeit ist nur Semikolon erlaubt.

Ausgabeformat ist optional. Wird *Ausgabeformat* nicht verwendet, liefert die Script-Funktion eine Zeit mit dem Defaultformat "HHMMSS" zurück.

Beispiele

Das erste Beispiel verwendet keine Zeitformate. Das Ergebnis ("130000") wird im Aktivierungsprotokoll ausgegeben.

```
:SET  &TIME# = ADD_TIME("120000","010000")
:PRINT  &TIME#
```

Das zweite Beispiel verwendet Zeitformat und Ausgabeformat. Das Ergebnis ist "04:59".

```
:SET  &TIME# = ADD_TIME("235959", "HH;05", "HH:MM")
:PRINT  &TIME#
```

Im dritten Beispiel werden zu 23:59:59 71 Sekunden addiert. Zeitformat wird verwendet, Ausgabeformat nicht. Das Ergebnis ("000110") entspricht dem Standardformat.

```
:SET  &TIME# = ADD_TIME("HH:MM:SS;23:59:59", "SS;71")
:PRINT  &TIME#
```

Gleiches Beispiel wie das Dritte. Aber im Unterschied dazu wird als Ausgabeformat ein Teilbegriff verwendet. Das Ergebnis ("70") wird in Sekunden im Aktivierungsprotokoll ausgegeben.

```
:SET  &TIME# = ADD_TIME("HHMMSS;235959", "SS;71", SS)
:PRINT  &TIME#
```

Siehe auch:

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.4 ADD_TIMESTAMP

Script-Funktion: Addiert Zeit zu einem Zeitstempel.

Syntax

ADD_TIMESTAMP*(Zeitstempel,Zeit)*

Syntaxteil	Beschreibung/Format
Zeitstempel	Zeitstempel aus Datum und Uhrzeit ("JJJJ-MM-TT HH:MM:SS"). Format: Script-Literal oder Script-Variable
Zeit	Zeit ("HH:MM:SS"), die zum Zeitstempel hinzugezählt werden soll. Format: Script-Literal oder Script-Variable Erlaubte Werte: 0 bis 99 (jeweils für HH, MM und SS)

Rückgabewert
Zeitstempel im Format "JJJJ-MM-TT HH:MM:SS"

Anmerkungen

Die Script-Funktion addiert Zeit im Format "HH:MM:SS" zu einem angegebenen Zeitstempel im Format "JJJJ-MM-TT HH:MM:SS".

Dabei wird mit UTC gerechnet, das heißt, Sommer- und Winterzeit werden nicht berücksichtigt. Da für Stunden, Minuten und Sekunden jeweils ein Maximalwert von 99 möglich ist, kann das Ergebnis der Script-Funktion etwas mehr als 4 Tage nach dem ursprünglichen Zeitstempel liegen.

Beispiel

Im Beispiel werden zum Zeitstempel 24 Stunden und eine Sekunde addiert. Das Ergebnis "2004-01-01 00:00:01" wird im Aktivierungsprotokoll ausgegeben.

```
:SET &RET# = ADD_TIMESTAMP("2003-12-31 00:00:00", "24:00:01")
 :PRINT &RET#
```

Siehe auch:

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.5 CALE_LOOK_AHEAD

Script-Funktion: Ermittelt das nächste Datum aufgrund von Kalenderbedingungen.

Syntax

CALE_LOOK_AHEAD(*[Datum]***, JOBP\JSCH, Aufgabennummer)**
CALE_LOOK_AHEAD(*[Datum]***, Bedingung, Kalender, Kalenderbegriff** *[[,Kalender] [,Kalenderbegriff]]...***)**

Syntaxteil	Beschreibung/Format
Datum	Angabe eines Datums im Format "JJMMTT" oder "JJJJMMTT". Format: Script-Literal oder Script-Variable Es ist möglich, das Datum in einem anderen Datumsformat anzugeben. Geben Sie dazu zuerst das gewünschte Datumsformat, ein Trennzeichen (: oder ;) und danach das Datum an. Die Angabe des Datumsformats ist optional.
JOBP\JSCH	Name eines Workflows oder eines Schedules. Format: Script-Literal oder Script-Variable
Aufgabennummer	Nummer der Aufgabe innerhalb des Workflows bzw. des Schedules. Format: Script-Literal oder Script-Variable

Bedingung	Bedingung, die bei der Datumsbestimmung zutreffen muss. Format: Script-Literal oder Script-Variable Erlaubte Werte: "ONE", "ALL", "NO" "ONE" - Eine der angegebenen Kalenderbedingungen muss zutreffen. "ALL" - Alle angegebenen Kalenderbedingungen müssen zutreffen. "NO" - Keine der angegebenen Kalenderbedingungen darf zutreffen. Trifft für eine Aufgabe keine Kalenderbedingung zu, wird ein Leerzeichen zurückgeliefert. Im Schedule-Monitor würde in diesem Fall CALE_LOOK_AHEAD_MAX angezeigt.
Kalender	Name eines Kalenders. Format: Script-Literal oder Script-Variable
Kalenderbegriff	Kalenderbegriff innerhalb dieses Kalenders Format: Script-Literal oder Script-Variable

Rückgabewerte
Datum im angegebenen Format. " " - Es trifft keine Kalenderbedingung zu.

Anmerkungen

Die Script-Funktion kann auf zwei unterschiedliche Arten verwendet werden.

Unter Angabe eines Workflows oder Schedules ist es zum einen möglich, das nächste Datum eines zugehörigen Objektes zu ermitteln, an welchem es aufgrund der definierten Kalenderbedingungen laufen wird. Das Objekt wird über die Aufgabennummer bestimmt. Diese wird beim Workflow in der graphischen Ansicht angezeigt und richtet sich nach der Reihenfolge, in der die Objekte dem Workflow hinzugefügt wurden. Beim Schedule entspricht die Nummerierung der Objekte der Reihenfolge in der Liste der Registerkarte "Schedule".

Die Script-Funktion unterstützt zum anderen die Berechnung des nächsten Datums, an dem eine, alle oder keine der Kalenderbedingungen zutreffen. Es können bis zu 5 Kalender mit ihren Kalenderbegriffen angegeben werden. Die maximale Anzahl von Tagen, die bei der Ermittlung des nächsten gültigen Datums berücksichtigt wird, kann der Administrator in der Variablen UC_ CLIENT_SETTINGS mit dem Key "CALE_LOOK_AHEAD_MAX" für jeden Mandanten festlegen.

Es kann ein Datum angegeben werden, ab einschließlich dem die Kalenderbedingungen geprüft werden. Die Angabe des Datumsformats ist dabei optional. Wird kein Datumsformat verwendet, muss das Datum in den Formaten "JJMMTT" oder "JJJJMMTT" angegeben werden. Als Trennzeichen zwischen Datumsformat und Datum sind Doppelpunkt oder Semikolon erlaubt. Geben Sie kein Datum an, so wird für die Berechnung das jeweils aktuelle Datum verwendet.

Das Datum, das die Script-Funktion ermittelt, wird entweder im Standardformat oder in dem Format, das Sie im ersten Parameter definieren, geliefert.

Beachten Sie, dass die Parameter *Kalender* und *Kalenderbegriff* immer gemeinsam angegeben werden müssen (siehe Beispiel unten)!

Beispiele

In dem folgenden Beispiel soll das nächste Datum ermittelt werden, an dem das Objekt mit der Aufgabennummer "3" im Workflow MAWI.TAG läuft.

```
:SET &DATUM# = CALE_LOOK_AHEAD("JJJJMMTT:20041010","MAWI.TAG","3")
```

Das zweite Beispiel bestimmt das nächste Datum, an dem alle übergebenen Kalenderbedingungen zutreffen.

```
:SET &DATUM# = CALE_LOOK_AHEAD
(,
"ALL"
,"FIRMENKALENDER","ARBEITSTAGE","FIRMENKALENDER","BEREITSCHAFT01")
```

Siehe auch:

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.6 CONV_DATE

Script-Funktion: Konvertiert das Format eines Datums.

Syntax

CONV_DATE(*Datum*[, *DatumsformatNeu*])

Syntaxteil	Beschreibung/Format
Datum	Angabe eines Datums im Format "JJMMTT" oder "JJJJMMTT". Format: Script-Literal oder Script-Variable Es ist möglich, das Datum in einem anderen Datumsformat anzugeben. Geben Sie dazu zuerst das gewünschte Datumsformat, ein Trennzeichen (: oder ;) und danach das Datum an. Die Angabe des Datumsformats ist optional.
DatumsformatNeu	Neue Formatvorgabe für das angegebene *Datum*. Format: Script-Literal oder Script-Variable

Rückgabewert
Datum in dem neuen Format

Anmerkungen

Die Script-Funktion konvertiert das Datumsformat von *Datum* in *DatumsformatNeu.*

Die Angabe des alten Datumformats und *DatumsformatNeu* sind optional. Sie dienen dazu das Format anzugeben, in dem die Funktion das *Datum* erhalten bzw. ihren Wert zurückgeben soll.

Wird das alte Datumsformat nicht verwendet, muss das Datum in den Formaten "JJMMTT" oder "JJJJMMTT" angegeben werden. Haben Sie *DatumsformatNeu* nicht angegeben, wird das Defaultformat "JJMMTT" zurückgegeben. Als Trennzeichen zwischen dem alten Datumsformat und *Datum* sind Doppelpunkt oder Semikolon erlaubt.

Beispiele

Die Beispiele konvertieren ein Datum von einem Datumsformat in ein anderes. Der von der Funktion zurückgegebene Wert wird in einer Script-Variablen hinterlegt. Dabei wird auch auf die Angabe von des alten Datumsformat bzw. *DatumsformatNeu* verzichtet. Die Rückgabewerte sind "31.12.1999", "31-12-1999" und "991231".

```
:SET  &STICHTAG# = CONV_DATE("TTMMJJ:311299", "TT.MM.JJJJ")
:SET  &STICHTAG# = CONV_DATE("991231", "TT-MM-JJJJ")
:SET  &STICHTAG# = CONV_DATE("TTMMJJ:311299")
```

Siehe auch:

Script-Sprachmittel	Beschreibung
CONV_TIMESTAMP	Rechnet Datum und Uhrzeit für eine andere Zeitzone um.

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.7 CONV_TIMESTAMP

Script-Funktion: Rechnet Datum und Uhrzeit für eine andere Zeitzone um.

Syntax

CONV_TIMESTAMP*(Zeitstempel,* *[Zeitzone1][,Zeitzone2]****)***

Syntaxteil	Beschreibung/Format
Zeitstempel	Zeitstempel aus Datum und Uhrzeit ("JJJJ-MM-TT HH:MM:SS"). Format: Script-Literal oder Script-Variable
Zeitzone1	Name eines Zeitzonen-Objektes oder Schlüsselwort UTC. Dem Zeitstempel zugeordnete Zeitzone oder UTC. Format: Script-Literal oder Script-Variable Standard: "UTC"
Zeitzone2	Name eines Zeitzonen-Objektes. Zeitzone, für die der Zeitstempel umgerechnet werden soll. Format: Script-Literal oder Script-Variable

Rückgabewert
Zeitstempel im Format "JJJJ-MM-TT HH:MM:SS".

Anmerkungen

Die Script-Funktion rechnet Datum und Uhrzeit im Format "JJJJ-MM-TT HH:MM:SS" für eine andere Zeitzone um.

Dabei kann mit *Zeitzone1* die ursprüngliche Zeitzone angegeben werden, von der die Berechnung ausgeht. Wird das Schlüsselwort UTC verwendet oder dieser Parameter ausgelassen, gilt UTC (Universal Time Coordinated) als Berechnungsgrundlage. Mit *Zeitzone2* wird die Zeitzone benannt, für die der Zeitstempel umgerechnet werden soll. Wird dieser optionale Parameter nicht angegeben, gilt die Zeitzone des Objektes bzw. die des Mandanten, wenn für das Objekt selbst keine Zeitzone definiert wurde.

Wird UTC nicht als Berechnungsgrundlage verwendet, ist die Umrechnung für die Zeit der Umstellung von Sommerzeit auf Normalzeit ungenau. Es kann nicht erkannt werden, ob beispielsweise mit 02:30:00 das erste oder zweite Auftreten dieser Uhrzeit gemeint ist. Die Automation Engine geht immer vom ersten Auftreten der Uhrzeit während der Umstellung auf Normalzeit aus.

Beispiel

Das Beispiel zeigt die Umrechnung des Zeitstempels für den Jahreswechsel 2003/2004. Ausgang der Berechnung ist eine für Sydney, Australien definierte Zeitzone, umgerechnet wird in mitteleuropäische Zeit. Das Ergebnis "2003-12-31 14:00:00" wird im Aktivierungsprotokoll ausgegeben.

```
:SET &MEZ# = CONV_TIMESTAMP("2004-01-01 00:00:00","TZ.SYD","TZ.MEZ")
:PRINT &MEZ#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
CONV_DATE	Konvertiert das Format eines Datums.

Script-Sprachmittel - Datum und Zeit
Zeit
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.8 DAY_OF_YEAR

Script-Funktion: Liefert den laufenden Tag eines Datums im Jahr.

Syntax

DAY_OF_YEAR*(Datum)*

Syntaxteil	Beschreibung/Format
Datum	Angabe eines Datums im Format "JJMMTT" oder "JJJJMMTT". Format: Script-Literal oder Script-Variable Es ist möglich, das Datum in einem anderen Datumsformat anzugeben. Geben Sie dazu zuerst das gewünschte Datumsformat, ein Trennzeichen (: oder ;) und danach das Datum an. Die Angabe des Datumsformats ist optional.

Rückgabewert
Laufender Tag des Datums im Jahr.

Anmerkungen

Die Angabe eines Datumsformats ist optional. Wird kein bestimmtes Datumsformat verwendet, muss das Datum in den Formaten "JJMMTT" oder "JJJJMMTT" angegeben werden. Als Trennzeichen zwischen Datumsformat und *Datum* sind Doppelpunkt oder Semikolon erlaubt.

Beispiel

Das Beispiel liefert den Wert "366" als laufenden Tag im Jahr.

```
:SET &LFDTAG# = DAY_OF_YEAR("TT.MM.JJ:31.12.00")
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_TIME_PHYSICAL	Stellt die jeweils aktuelle Tageszeit fest.
SYS_DATE_PHYSICAL	Ermittelt das jeweils aktuelle Tagesdatum.
SYS_TIMESTAMP_PHYSICAL	Liefert aktuelles Datum und Uhrzeit.

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.9 DIFF_DATE

Script-Funktion: Ermittelt die Differenz zwischen zwei Datumsangaben in Tagen.

Syntax

DIFF_DATE*(Datum1,Datum2)*

Syntaxteil	Beschreibung/Format

Datum1 und *Datum2*	Angabe eines Datums im Format "JJMMTT" oder "JJJJMMTT". Format: Script-Literal oder Script-Variable Es ist möglich, das Datum in einem anderen Datumsformat anzugeben. Geben Sie dazu zuerst das gewünschte Datumsformat, ein Trennzeichen (: oder ;) und danach das Datum an. Die Angabe des Datumsformats ist optional.

Rückgabewert

Anzahl an Tagen, die zwischen den beiden Datumsangaben liegt.

Anmerkungen

Die Script-Funktion ermittelt die Differenz zwischen *Datum1* und *Datum2* in Tagen. Dabei kann *Datum2* jünger oder älter als *Datum1* sein.

Bei den Datumsangaben ist das Datumsformat optional. Wird kein bestimmtes Datumsformat verwendet, muss das Datum in den Formaten "JJMMTT" oder "JJJJMMTT" angegeben werden. Als Trennzeichen zwischen Datumsformat und Datum sind Doppelpunkt oder Semikolon erlaubt.

Beispiele

Das erste Beispiel ermittelt die Differenz zwischen den Datumsangaben. Die Daten verwenden die Standardformate "JJMMTT" und "JJJJMMTT". Das Ergebnis "1" wird im Report ausgegeben.

```
:SET &DIFF# = DIFF_DATE("000330","20000331")
 :PRINT &DIFF#
```

Im zweiten Beispiel werden die Datumsangaben den Script-Variablen zugewiesen. Die Script-Funktion wird mit diesen Script-Variablen aufgerufen. Da die Daten kein Standardformat verwenden, muss auch bei den Script-Variablen das spezielle Datumsformat angegeben werden. Das Ergebnis "366" wird im Report ausgegeben.

```
:SET &DATUM1# = "01-01-2000"
 :SET &DATUM2# = "01012001"
 :SET &DIFF# = DIFF_DATE("DD-MM-YYYY:&DATUM1#", "TTMMJJJJ;&DATUM2#")
 :PRINT &DIFF#
```

Das dritte Beispiel hat das gleiche Ergebnis wie das zweite Beispiel. Der Unterschied zwischen beiden besteht darin, dass hier den Script-Variablen Datumsformat plus Datum zugewiesen wird.

```
:SET &DATUM1# = "TT-MM-JJJJ:01-01-2000"
 :SET &DATUM2# = "TTMMJJJJ:01012001"
 :SET &DIFF# = DIFF_DATE(&DATUM1#, &DATUM2#)
 :PRINT &DIFF#
```

Siehe auch:

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.10 DIFF_TIME

Script-Funktion: Ermittelt die Differenz zwischen zwei Zeitangaben.

DIFF_TIME*(Zeit1, Zeit2 *[,Ausgabeformat]**)***

Syntaxteil	Beschreibung/Format
Zeit1 und *Zeit2*	Angabe einer Zeit im Format "HHMMSS". Format: Script-Literal oder Script-Variable Es ist möglich, die Zeit in einem anderen Zeitformat anzugeben. Geben Sie dazu zuerst das gewünschte Zeitformat, ein Trennzeichen (;) und danach die Zeit an. Die Angabe des Zeitformats ist optional.
Ausgabeformat	Formatvorgabe für die ermittelte Zeit. Format: AE-name, Script-Literal oder Script-Variable Standard: "HHMMSS"

Rückgabewert
Zeit im angegebenen Format.

Anmerkungen

Die Script-Funktion ermittelt die Differenz zwischen *Zeit1* und *Zeit2*. Es wird immer der Abstand von der jeweils kleineren zur größeren Zeit errechnet. Dabei ist unerheblich, ob die größere Zeit mit *Zeit1* oder *Zeit2* angegeben wurde.

Der Script-Funktion werden zwei Zeiten übergeben. Die eines Zeitformats ist dabei optional. Wird kein bestimmtes Zeitformat verwendet, muss die Zeit im Format "HHMMSS" angegeben werden. Als Trennzeichen zwischen Zeitformat und Zeit ist nur Semikolon erlaubt. Werden Teilbegriffe beim Zeitformat verwendet, wird mit der angegebenen Anzahl von Stunden, Minuten oder Sekunden gerechnet. Für die fehlenden Informationen wird jeweils "00" gesetzt.

Ausgabeformat ist optional. Wird *Ausgabeformat* nicht verwendet, liefert die Script-Funktion eine Zeit mit dem Standardformat "HHMMSS" zurück.

Beispiele

Das erste Beispiel ermittelt die Differenz zwischen den Zeitangaben. Die Zeiten verwenden das Standardformat "HHMMSS". Das Ergebnis "003000" wird im Report ausgegeben.

```
:SET  &DIFF# = DIFF_TIME("230000", "223000")
:PRINT  &DIFF#
```

In diesem Beispiel wird bei der zweiten Zeit ein Teilbegriff verwendet. Damit sind deren Stunden und Sekunden automatisch "00". Ein Ausgabeformat wird vorgegeben. Das Ergebnis "00:09" erscheint im Report.

```
:SET  &DIFF# = DIFF_TIME("HH:MM:SS;00:01:30", "MM;11", "HH:MM")
:PRINT  &DIFF#
```

Im dritten Beispiel werden die Zeiten den Script-Variablen zugewiesen. Die Script-Funktion wird mit diesen Script-Variablen aufgerufen. Da die Zeiten kein Standardformat verwenden, muss auch bei

den Script-Variablen das spezielle Zeitformat angegeben werden. Das Ergebnis "120000" wird im Report ausgegeben.

```
:SET  &ZEIT1# = "00:00:00"
:SET  &ZEIT2# = "12:00:00"
:SET  &DIFF# = DIFF_TIME("HH:MM:SS;&ZEIT1#", "HH:MM:SS;&ZEIT2#")
:PRINT  &DIFF#
```

Siehe auch:

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.11 FIRST_OF_PERIOD

Script-Funktion: Ermittelt den ersten Tag der Periode für ein angegebenes Datum.

Syntax

FIRST_OF_PERIOD*(Datum, Periodenformat[, Ausgabeformat[, Kalender, Kalenderbegriff]])*

Syntaxteil	Beschreibung/Format
Datum	Angabe eines Datums im Format "JJMMTT" oder "JJJJMMTT". Format: Script-Literal oder Script-Variable Es ist möglich, das Datum in einem anderen Datumsformat anzugeben. Geben Sie dazu zuerst das gewünschte Datumsformat, ein Trennzeichen (: oder ;) und danach das Datum an. Die Angabe des Datumsformats ist optional.
Periodenformat	Angabe eines gültigen Periodenformates. Format: Script-Literal oder Script-Variable
Ausgabeformat	Formatvorgabe für das ermittelte Datum. Format: Script-Literal oder Script-Variable Standard: JJMMTT
Kalender	Name des Kalenders, der für den Periodenbeginn berücksichtigt werden soll. Format: AE-name, Script-Literal oder Script-Variable
Kalenderbegriff	Name des Kalenderbegriffs, der für den Periodenbeginn berücksichtigt werden soll. Format: AE-Name, Script-Literal oder Script-Variable

Rückgabewerte

Datum des Periodenbeginns im angegebenen Format.
"20327" - Das Kalender-Objekt existiert nicht.
"20328" - Der Kalenderbegriff existiert nicht im Kalender-Objekt.
"20456" - Im Kalenderbegriff ist das Startdatum der Periode nicht enthalten.

Anmerkungen

Die Script-Funktion ermittelt den ersten Tag der Periode, zu der ein angegebenes Datum gehört.

Der Script-Funktion wird ein Datum übergeben. Die Angabe des Datumsformats ist dabei optional. Wird kein bestimmtes Datumsformat verwendet, muss das Datum in den Formaten "JJMMTT" oder "JJJJMMTT" angegeben werden. Als Trennzeichen zwischen Datumsformat und *Datum* sind Doppelpunkt oder Semikolon erlaubt.

Mit *Periodenformat* wird die Art der Periode (Jahr, Quartal, Monat oder Woche) festgelegt

Für diese Script-Funktion gibt es das spezielle Periodenformat "WS". Damit wird als erster Tag einer Woche immer der Sonntag genommen. "WW" hingegen geht stets vom Montag als ersten Tag der Woche aus.

Ausgabeformat ist optional. Wird *Ausgabeformat* nicht verwendet, liefert die Script-Funktion ein Datum mit dem Defaultformat "JJMMTT" zurück. Bitte beachten Sie, dass Kommata auch dann zu setzen sind, wenn kein *Ausgabeformat* verwendet wird, aber *Kalender* und *Kalenderbegriff* angegeben werden.

Werden *Kalender* und *Kalenderbegriff* verwendet, gibt die Script-Funktion den ersten zutreffenden Kalendertag der Periode zurück. Gibt es keinen gültigen Kalendertag für diese Periode, ist das Ergebnis ein Null-Datum im jeweiligen Datumsformat (z.B.: 0000-00-00).

Mit der Script-Anweisung :ON_ERROR legen Sie die Reaktion auf diesen Fehler fest. Wie bisher können Sie ihn mit den Script-Funktionen zur Fehlerbehandlung analysieren. Das Script wird weiter prozessiert. Es besteht aber auch die Möglichkeit, die Prozessierung des Scripts abzubrechen.

Beispiele

Das erste Beispiel ermittelt, den ersten Tag der Woche für den 29.3.2000. Als Ergebnis wird der 27.03.00 (Montag) im Aktivierungsprotokoll ausgegeben.

```
:SET &DATUM# = FIRST_OF_PERIOD ("000329","WW","TT.MM.JJ")
:PRINT &DATUM#
```

Mit dem speziellen Periodenformat "WS" steht der 26.03.00 (Sonntag) als Ergebnis im Aktivierungsprotokoll.

```
:SET &DATUM# = FIRST_OF_PERIOD ("TT.MM.JJ:29.03.00","WS","TT.MM.JJ")
:PRINT &DATUM#
```

Im dritten Beispiel wird der erste zutreffende Kalendertag eines Quartals ermittelt. Auf die Vorgabe eines Ausgabeformates wird dabei verzichtet.

```
:SET &DATUM# = FIRST_OF_PERIOD ("000329", "Q", , BEREITSCHAFT,
WOCHENTAG)
:PRINT &DATUM#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
LAST_OF_PERIOD	Ermittelt den letzten Tag der Periode für ein angegebenes Datum.
:ON_ERROR	Legt die Reaktion auf bestimmte Fehler bzw. Meldungen von Script-Sprachmitteln fest.

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.12 LAST_OF_PERIOD

Script-Funktion: Ermittelt den letzten Tag der Periode für ein angegebenes Datum.

Syntax

LAST_OF_PERIOD*(Datum, Periodenformat[, *Ausgabeformat*[, *Kalender*, *Kalenderbegriff*]]***)***

Syntaxteil	Beschreibung/Format
Datum	Angabe eines Datums im Format "JJMMTT" oder "JJJJMMTT". Format: Script-Literal oder Script-Variable Es ist möglich, das Datum in einem anderen Datumsformat anzugeben. Geben Sie dazu zuerst das gewünschte Datumsformat, ein Trennzeichen (: oder ;) und danach das Datum an. Die Angabe des Datumsformats ist optional.
Periodenformat	Angabe eines gültigen Periodenformates. Format: Script-Literal oder Script-Variable
Ausgabeformat	Formatvorgabe für das ermittelte Datum. Format: Script-Literal oder Script-Variable Standard: JJMMTT
Kalender	Name des Kalenders, der für das Perioden-Ende berücksichtigt werden soll. Format: AE-name, Script-Literal oder Script-Variable
Kalenderbegriff	Name des Kalenderbegriffs, der für das Perioden-Ende berücksichtigt werden soll. Format: AE-Name, Script-Literal oder Script-Variable

Rückgabewerte
Datum des Periodenendes im angegebenen Format. "20327" - Das Kalender-Objekt existiert nicht. "20328" - Der Kalenderbegriff existiert nicht im Kalender-Objekt. "20456" - Im Kalenderbegriff ist das Enddatum der Periode nicht enthalten.

Anmerkungen

Die Script-Funktion ermittelt den letzten Tag der Periode, zu der ein angegebenes Datum gehört. Der Rückgabewert ist ein Datum.

Der Script-Funktion wird ein Datum übergeben. Die Angabe eines Datumsformats ist dabei optional. Wird kein bestimmtes Datumsformat verwendet, muss das Datum in den Formaten "JJMMTT" oder "JJJJMMTT" angegeben werden. Als Trennzeichen zwischen Datumsformat und *Datum* sind Doppelpunkt oder Semikolon erlaubt.

Mit *Periodenformat* wird die Art der Periode (Jahr, Quartal, Monat oder Woche) festgelegt. Hinweis: Für diese Script-Funktion gibt es das spezielle Periodenformat "WS". Damit wird als letzter Tag einer Woche immer der Samstag genommen. "WW" hingegen geht stets vom Sonntag als letzten Tag der Woche aus.

Ausgabeformat ist optional. Wird *Ausgabeformat* nicht verwendet, liefert die Script-Funktion ein Datum mit dem Standardformat "JJMMTT" zurück. Bitte beachten Sie: Kommata sind auch zu setzen, wenn *Ausgabeformat* nicht verwendet wird, aber *Kalender* und *Kalenderbegriff* angegeben werden.

Werden *Kalender* und *Kalenderbegriff* verwendet, gibt die Script-Funktion den letzten zutreffenden Kalendertag der Periode zurück. Gibt es keinen gültigen Kalendertag für diese Periode, ist das Ergebnis ein Null-Datum im jeweiligen Datumsformat (z.B.: 0000-00-00).

Mit der Script-Anweisung :ON_ERROR legen Sie die Reaktion auf diesen Fehler fest. Wie bisher können Sie ihn mit den Script-Funktionen zur Fehlerbehandlung analysieren. Das Script wird weiter prozessiert. Es besteht aber auch die Möglichkeit, die Prozessierung des Scripts abzubrechen.

Beispiele

Das erste Beispiel ermittelt, den letzten Tag der Woche für den 29.3.2000. Als Ergebnis wird der 02.04.00 (Sonntag) im Aktivierungsprotokoll ausgegeben.

```
:SET  &DATUM# = LAST_OF_PERIOD ("000329","WW","TT.MM.JJ")
:PRINT  &DATUM#
```

Mit dem speziellen Periodenformat "WS" steht der 01.04.00 (Samstag) als Ergebnis im Aktivierungsprotokoll.

```
:SET  &DATUM# = LAST_OF_PERIOD ("TT.MM.JJ:29.03.00","WS","TT.MM.JJ")
:PRINT  &DATUM#
```

Im dritten Beispiel wird der letzte zutreffende Kalendertag eines Quartals ermittelt. Auf die Vorgabe eines Ausgabeformates wird dabei verzichtet.

```
:SET  &DATUM# = LAST_OF_PERIOD ("000329", "Q", , BEREITSCHAFT,
WOCHENTAG)
:PRINT  &DATUM#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
FIRST_OF_PERIOD	Ermittelt den ersten Tag der Periode für ein angegebenes Datum.

:ON_ERROR	Legt die Reaktion auf bestimmte Fehler bzw. Meldungen von Script-Sprachmitteln fest.

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.13 SUB_DAYS

Script-Funktion: Subtrahiert Tage von angegebenem Datum.

Syntax

SUB_DAYS*(Datum, Tage[, Kalender, Kalenderbegriff])*

Syntaxteil	Beschreibung/Format
Datum	Angabe eines Datums im Format "JJMMTT" oder "JJJJMMTT". Format: Script-Literal oder Script-Variable Es ist möglich, das Datum in einem anderen Datumsformat anzugeben. Geben Sie dazu zuerst das gewünschte Datumsformat, ein Trennzeichen (: oder ;) und danach das Datum an. Die Angabe des Datumsformats ist optional.
Tage	Anzahl von Tagen, mit denen gerechnet werden soll. Format: Script-Literal, Zahl ohne Hochkommata oder Script-Variable
Kalender	Name des Kalenders, der bei Berechnung zu berücksichtigen ist. Format: AE-name, Script-Literal oder Script-Variable
Kalenderbegriff	Name des Kalenderbegriffes, der bei Berechnung zu berücksichtigen ist. Format: AE-Name, Script-Literal oder Script-Variable

Rückgabewert
Datum im angegebenen Format "0" - Dem Kalenderbegriff wurden keine Tage zugewiesen oder das berechnete Datum liegt außerhalb des Kalenders. "20327" - Das Kalender-Objekt existiert nicht. "20328" - Der Kalenderbegriff existiert nicht im Kalender-Objekt.

Anmerkungen

Mit der Script-Funktion subtrahieren Sie *Tage* von einem angegebenen *Datum*. Optional können Sie diese Rechenoperationen auch in Abhängigkeit von einem *Kalender* und einem *Kalenderbegriff* ausführen. Es werden dann nur solche *Tage* berücksichtigt, die laut Definition im *Kalender* gültig sind.

Wird ein bestimmtes Datumsformat angegeben, entspricht das zurückgegebene Datum diesem Format. Wird kein Datumsformat verwendet, muss das Datum in den Formaten "JJMMTT" oder

"JJJJMMTT" angegeben werden. Es wird im selben Format zurückgegeben. Als Trennzeichen zwischen Datumsformat und *Datum* sind Doppelpunkt oder Semikolon erlaubt.

Insbesondere sei auf die Möglichkeit hingewiesen, die Anzahl der *Tage* gleich Null zu setzen. Wenn das angegebene *Datum* in den *Kalenderbegriff* fällt, wird das *Datum* selbst zurückgegeben, ansonsten das vorhergehende gültige *Datum* laut *Kalenderbegriff*.

⚠ Bei der Datumsberechnung wird der Bereich des angegebenen Kalenders berücksichtigt. Dieser hängt von der Einstellung ab, die der Administrator in der Variablen UC_CLIENT_SETTINGS mit den Keys NOW_MINUS und NOW_PLUS konfiguriert hat. Fällt das errechnete Datum außerhalb des Kalenders, liefert die Script-Funktion den Wert "0" zurück.

Beispiele

Das erste Beispiel subtrahiert zwei Tage von einem angegebenen Datum. Das Ergebnis ("000330") wird im Aktivierungsprotokoll ausgegeben.

```
:SET &DATUM# = SUB_DAYS("000401", 2)
:PRINT &DATUM#
```

Das zweite Beispiel erfragt durch eine Script-Anweisung die Anzahl von Tagen vom Benutzer, die aufzuheben ist. Mit der Funktion wird diese Anzahl vom aktuellen Tagesdatum abgezogen und in einer Script-Variablen hinterlegt. Da kein Datumsformat angegeben wird, verwendet jedes Datum das Defaultformat "JJMMTT".

```
:SET &AKTDAT# = SYS_DATE()
:READ &ANZAHL#,"00","Wieviele Tage aufzuheben?","08",N
:SET &RETDAT# = SUB_DAYS(&AKTDAT#, &ANZAHL#)
```

Im dritten Beispiel wird das aktuelle Datum ermittelt. Davon werden anschließend zwei Tage abgezogen. Da das aktuelle Tagesdatum nicht im Standardformat abgefragt wurde, muss auch bei der Script-Variablen das spezielle Datumsformat angegeben werden.

```
:SET &AKTDAT# = SYS_DATE("TT.MM.JJ")
:SET &VARBTAG# = SUB_DAYS("TT.MM.JJ:&AKTDAT#", 2)
```

Siehe auch:

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.14 SUB_PERIOD

Script-Funktion: Subtrahiert eine Periode von einem angegebenen Datum.

Syntax

SUB_PERIOD*(Datum, Periodenformat:Periode*[*, Ausgabeformat*]*)*

Syntaxteil	Beschreibung/Format

Datum	Angabe eines Datums im Format "JJMMTT" oder "JJJJMMTT". Format: Script-Literal oder Script-Variable Es ist möglich, das Datum in einem anderen Datumsformat anzugeben. Geben Sie dazu zuerst das gewünschte Datumsformat, ein Trennzeichen (: oder ;) und danach das Datum an. Die Angabe des Datumsformats ist optional.
Periodenformat:Periode	Format: Script-Literal oder Script-Variable Periode: Angabe einer Periode, die zum angegebenen Datum addiert werden soll. Periodenformat: Formatvorgabe für die *Periode*. Als Trennzeichen zwischen dem *Periodenformat* und der *Periode* ist ein Doppelpunkt (:) oder ein Semikolon (;) erlaubt.
Ausgabeformat	Formatvorgabe für das ermittelte Datum. Format: AE-name, Script-Literal oder Script-Variable Standardwert: "JJMMTT"

Rückgabewert
Datum im angegebenen Format.

Anmerkungen

Die Script-Funktion zieht Jahre, Monate, Quartale oder Wochen von einem angegebenen Datum ab.

Der Script-Funktion wird ein Datum übergeben. Die Angabe eines Datumsformats ist dabei optional. Wird kein Datumsformat verwendet, muss das Datum in den Formaten "JJMMTT" oder "JJJJMMTT" angegeben werden. Als Trennzeichen zwischen Datumsformat und *Datum* sind Doppelpunkt oder Semikolon erlaubt.

Vom angegebenen *Datum* wird die *Periode* abgezogen. *Periode* kann eine beliebige Zahl sein. Ist das Ergebnis ein Datum, welches vor dem Jahr 0000 liegen würde, führt das zu einem Fehler.

Ausgabeformat ist optional. Wird *Ausgabeformat* nicht verwendet, liefert die Script-Funktion ein Datum mit dem Standardformat "JJMMTT" zurück.

Beispiele

Das erste Beispiel subtrahiert zwei Wochen vom 5.7.2000. Das Ergebnis (21.06.2000) wird im Report ausgegeben.

```
:SET  &DATUM#=SUB_PERIOD ("TT.MM.JJ:05.07.00","WW;2","TT.MM.JJJJ")
:PRINT  &DATUM#
```

Das zweite Beispiel subtrahiert ein Quartal vom 31.7.2000. Das Ergebnis (30-04-2000) wird im Report ausgegeben.

```
:SET  &DATUM#=SUB_PERIOD ("000731","Q:1","TT-MM-JJJJ")
:PRINT  &DATUM#
```

Im dritten Beispiel wird ein Jahr vom 28.2.2001 subtrahiert. Als Ergebnis wird 28.02.2000 im Report ausgegeben.

```
:SET  &DATUM#=SUB_PERIOD ("20010228","JJ:0001",TT.MM.JJJJ)
:PRINT  &DATUM#
```

Siehe auch:

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.15 SUB_TIME

Script-Funktion: Subtrahiert zwei Zeiten.

Syntax

SUB_TIME*(Zeit1, Zeit2 *[,Ausgabeformat]**)***

Syntaxteil	Beschreibung/Format
Zeit1 und *Zeit2*	Angabe einer Zeit im Format "HHMMSS". Format: Script-Literal oder Script-Variable Es ist möglich, die Zeit in einem anderen Zeitformat anzugeben. Geben Sie dazu zuerst das gewünschte Zeitformat, ein Trennzeichen (;) und danach die Zeit an. Die Angabe des Zeitformats ist optional.
Ausgabeformat	Formatvorgabe für die ermittelte Zeit. Format: AE-name, Script-Literal oder Script-Variable Standard: "HHMMSS"

Rückgabewert
Zeit im angegebenen Format

Anmerkungen

Mit der Script-Funktion subtrahieren Sie zwei Zeiten. Dabei wird der Wechsel von 00:00:00 auf 23:59:59 berücksichtigt.

Der Script-Funktion werden zwei Zeiten übergeben. Die Angabe eines Zeitformats ist dabei optional. Wird kein bestimmtes Zeitformat verwendet, muss die Zeit im Format "HHMMSS" angegeben werden. Als Trennzeichen zwischen Zeitformat und Zeit ist nur Semikolon erlaubt.

Ausgabeformat ist optional. Wird *Ausgabeformat* nicht verwendet, liefert die Script-Funktion eine Zeit mit dem Standardformat "HHMMSS" zurück.

Beispiele

Das erste Beispiel verwendet keine Zeitformate. Das Ergebnis ("110000") wird im Aktivierungsprotokoll ausgegeben.

```
:SET  &TIME# = SUB_TIME("120000","010000")
:PRINT  &TIME#
```

Das zweite Beispiel verwendet Zeitformat und Ausgabeformat. Das Ergebnis ist "23:00".

```
:SET  &TIME# = SUB_TIME("040000", "HH;05", "HH:MM")
:PRINT  &TIME#
```

Im dritten Beispiel werden von 00:00:10 31 Sekunden subtrahiert. Zeitformat wird verwendet, Ausgabeformat nicht. Das Ergebnis ("235940") entspricht dem Standardformat.

```
:SET  &TIME# = SUB_TIME("HH:MM:SS;00:00:10", "SS;30")
:PRINT  &TIME#
```

Gleiches Beispiel wie das dritte. Aber im Unterschied dazu wird als Ausgabeformat ein Teilbegriff verwendet. Das Ergebnis ("86380") entsteht durch Umrechnung von 23:59:40 in Sekunden.

```
:SET  &TIME# = SUB_TIME("HHMMSS;000010", "SS;30", SS)
:PRINT  &TIME#
```

Siehe auch:

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.16 SUB_TIMESTAMP

Script-Funktion: Subtrahiert Zeit von einem Zeitstempel.

Syntax

SUB_TIMESTAMP *(Zeitstempel, Zeit)*

Syntaxteil	Beschreibung/Format
Zeitstempel	Zeitstempel aus Datum und Uhrzeit ("JJJJ-MM-TT HH:MM:SS"). Format: Script-Literal oder Script-Variable
Zeit	Zeit ("HH:MM:SS"), die vom Zeitstempel abgezogen werden soll. Format: Script-Literal oder Script-Variable Erlaubte Werte: 0 bis 99 (jeweils für HH, MM und SS)

Rückgabewert
Zeitstempel im Format "JJJJ-MM-TT HH:MM:SS"

Anmerkungen

Die Script-Funktion subtrahiert Zeit im Format "HH:MM:SS" von einem angegebenen Zeitstempel im Format "JJJJ-MM-TT HH:MM:SS".

Dabei wird mit UTC gerechnet, das heißt, Sommer- und Winterzeit werden nicht berücksichtigt. Da für Stunden, Minuten und Sekunden ein Maximalwert von 99 möglich ist, kann das Ergebnis der Script-Funktion etwas mehr als 4 Tage vor dem ursprünglichen Zeitstempel liegen.

Beispiel

Im Beispiel wird vom Zeitstempel eine Sekunde abgezogen. Das Ergebnis "2003-12-31 23:59:59" wird im Aktivierungsprotokoll ausgegeben.

```
:SET &RET# = SUB_TIMESTAMP("2004-01-01 00:00:00", "00:00:01")
 :PRINT &RET#
```

Siehe auch:

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.17 SYS_DATE

Script-Funktion: Ermittelt das aktuelle Tagesdatum zu Beginn der Script-Prozessierung.

Syntax

SYS_DATE(*[Datumsformat][,Zeitzone]***)**

Syntaxteil	Beschreibung/Format
Datumsformat	Formatvorgabe für das ermittelte Datum. Format: Script-Literal oder Script-Variable Standard: "JJMMTT"
Zeitzone	Name eines Zeitzonen-Objektes oder Schlüsselwort UTC. Format: Script-Literal oder Script-Variable

Rückgabewert
Aktuelles Tagesdatum im angegebenen Format.

Anmerkungen

Die Script-Funktionen ermittelt das aktuelle Tagesdatum. Dabei wird eine Zeitzone, welche als Parameter angegeben wurde, berücksichtigt.

Datumsformat ist optional. Es dient dazu, das Format festzulegen, in dem die Funktion ihren Wert zurückgeben soll. Haben Sie *Datumsformat* nicht angegeben, wird das Standardformat "JJMMTT" zurückgegeben.

Zeitzone ist ebenfalls optional. Wird die Script-Funktion ohne diesen Parameter aufgerufen, wird die Zeitzone des Objektes verwendet bzw. die des Mandanten, wenn für das Objekt selbst keine Zeitzone definiert wurde. Wurde eine nicht definierte Zeitzone angegeben, wird automatisch mit dem Standardwert (Zeitzone des Mandanten) gerechnet. Anstatt einer Zeitzone kann auch das Schlüsselwort UTC verwendet werden. Das Datum wird direkt in UTC (Universal Time Coordinated) zurückgegeben.

Das aktuelle Tagesdatum wird zu Beginn der Script-Prozessierung ermittelt und "eingefroren". Damit wird die Konsistenz des Scripts sichergestellt. Wenn Sie die Script-Funktion innerhalb eines Scripts mehrfach verwenden, wird also immer dasselbe Tagesdatum zurückgegeben. Das gilt insbesondere auch dann, wenn die Scriptbearbeitung durch eine :WAIT-Anweisung für einen festgelegten Zeitraum unterbrochen wird.

Um das ermittelte Tagesdatum mit der Script-Anweisung :PUT_VAR an ein Variablen-Objekt vom Typ "Zeitstempel" zu übergeben, müssen die Datumsformate "JJMMTT" (Standard), "JJJJMMTT" oder "JJJJ-MM-TT" verwendet werden. Mit dem Speichern in der Variable geht dieses Datumsformat auf der Windows-Plattform verloren. Die Anzeige des Datums basiert nun auf den Ländereinstellungen in der Systemsteuerung von Windows.

Beispiele

Das erste Beispiel ermittelt das Tagesdatum und übergibt den Wert an eine Script-Variable. Im zweiten Beispiel soll gezeigt werden, dass auch die Angabe von Teilbegriffen möglich ist. Es wird der Wochentag festgestellt und als Funktionsparameter eine Script-Variable verwendet.

```
:SET &DATUM# = SYS_DATE("TT.MM.JJJJ")

:SET &FORMAT# = "WW"
 :SET &WOCHENTAG# = SYS_DATE(&FORMAT#)
```

Wird die Funktion ohne die Angabe von *Datumsformat* verwendet, ist die Syntax wie im folgenden Beispiel.

```
:IF SYS_DATE() = "990101"
!...
 :ENDIF
```

Im dritten Beispiel wird das aktuelle Tagesdatum ermittelt und in einem Variablen-Objekt vom Typ "Zeitstempel" gespeichert. Dabei wird eine Zeitzone berücksichtigt, die für mitteleuropäische Zeit definiert wurde.

```
:SET &DATUM# = SYS_DATE("JJJJ-MM-TT","TZ.MEZ")
 :PUT_VAR BUCHUNGSDATUM, , &DATUM#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
CONV_DATE	Konvertiert das Format eines Datums.
DIFF_DATE	Ermittelt die Differenz zwischen zwei Datumsangaben in Tagen.
SYS_DATE_PHYSICAL	Ermittelt das jeweils aktuelle Tagesdatum.
SYS_LDATE	Ermittelt das logische Datum.

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.18 SYS_DATE_PHYSICAL

Script-Funktion: Ermittelt das jeweils aktuelle Tagesdatum.

Syntax

SYS_DATE_PHYSICAL(*[Datumsformat][,Zeitzone]***)**

Syntaxteil	Beschreibung/Format
Datumsformat	Formatvorgabe für das ermittelte Datum. Format: Script-Literal oder Script-Variable Standardwert: "JJMMTT"
Zeitzone	Name eines Zeitzonen-Objektes oder Schlüsselwort UTC. Format: Script-Literal oder Script-Variable

Rückgabewert
Aktuelles Tagesdatum im angegebenen Format.

Anmerkungen

Die Script-Funktionen ermittelt das jeweils aktuelle Tagesdatum. Dabei wird eine Zeitzone, welche als Parameter angegeben wurde, berücksichtigt.

⚠ Wird die Script-Funktion mehrmals in einem Script verwendet, kann das unter Umständen zu unterschiedlichen Ergebnissen führen (zum Beispiel Datumswechsel zwischen erstem und zweiten Aufruf). Damit unterscheidet sich die Script-Funktion vonSYS_DATE, welche das aktuelle Tagesdatum zu Beginn der Script-Prozessierung ermittelt und dann "einfriert", um die Konsistenz des Scripts zu gewährleisten.

Datumsformat ist optional. Es dient dazu, das Format festzulegen, in dem die Funktion ihren Wert zurückgeben soll. Haben Sie *Datumsformat* nicht angegeben, wird das Standardformat "JJMMTT" zurückgegeben.

Zeitzone ist ebenfalls optional. Wird die Script-Funktion ohne diesen Parameter aufgerufen, wird die Zeitzone des Objektes verwendet bzw. die des Mandanten, wenn für das Objekt selbst keine Zeitzone definiert wurde. Wurde eine nicht definierte Zeitzone angegeben, wird automatisch mit dem Standardwert (Zeitzone des Mandanten) gerechnet. Anstatt einer Zeitzone kann auch das Schlüsselwort UTC verwendet werden. Das Datum wird direkt in UTC (Universal Time Coordinated) zurückgegeben.

Beispiele

Das erste Beispiel ermittelt das Tagesdatum und übergibt den Wert an eine Script-Variable. Im zweiten Beispiel soll gezeigt werden, dass auch die Angabe von Teilbegriffen möglich ist. Es wird der Wochentag festgestellt und als Funktionsparameter eine Script-Variable verwendet.

```
:SET &DATUM# = SYS_DATE_PHYSICAL('TT.MM.JJJJ')

:SET &FORMAT# = 'WW'
:SET &WOCHENTAG# = SYS_DATE_PHYSICAL(&FORMAT#)
```

Wird die Funktion ohne die Angabe von *Datumsformat* verwendet, ist die Syntax wie im folgenden Beispiel.

```
:IF SYS_DATE_PHYSICAL() = '990101'
!...
:ENDIF
```

Im dritten Beispiel wird das aktuelle Tagesdatum ermittelt. Dabei wird eine Zeitzone berücksichtigt, die für mitteleuropäische Zeit definiert wurde. Das Ergebnis wird im Aktivierungsprotokoll ausgegeben.

```
:SET &DATUM# = SYS_DATE_PHYSICAL('JJJJ-MM-TT','TZ.MEZ')
:PRINT &DATUM#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
CONV_DATE	Konvertiert das Format eines Datums.
DIFF_DATE	Ermittelt die Differenz zwischen zwei Datumsangaben in Tagen.
SYS_LDATE	Ermittelt das logische Datum.
SYS_DATE	Ermittelt das aktuelle Tagesdatum zu Beginn der Script-Prozessierung.

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.19 SYS_LDATE

Script-Funktion: Ermittelt das logische Datum.

Syntax

SYS_LDATE(*[Datumsformat]***)**

Syntaxteil	Beschreibung/Format
Datumsformat	Formatvorgabe für das ermittelte Datum. Format: Script-Literal oder Script-Variable Standardwert: "JJMMTT"

Rückgabewert
Das logische Datum im angegebenen Format.

Anmerkungen

Beim Ausführen von Aufgaben mit Optionen können Sie ein sogenanntes logisches Datum angeben. Die Script-Funktion ermittelt jenes.

Datumsformat ist optional. Es dient dazu das Format festzulegen, in dem die Funktion ihren Wert zurückgeben soll. Haben Sie *Datumsformat* nicht angegeben, wird das Standardformat "JJMMTT" zurückgegeben.

⚠ Beachten Sie, dass die Script-Funktion in !Scripts von Ereignissen nicht das logische, sondern das aktuelle Datum liefert!

Beispiel

Im folgenden Beispiel wird das logische Datum im Format "TT.MM.JJJJ" ermittelt.

```
:SET &LDATE# = SYS_LDATE("TT.MM.JJJJ")
```

Siehe auch:

Script-Sprachmittel	Beschreibung
CONV_DATE	Konvertiert das Format eines Datums.
DIFF_DATE	Ermittelt die Differenz zwischen zwei Datumsangaben in Tagen.
SYS_DATE	Ermittelt das aktuelle Tagesdatum zu Beginn der Script-Prozessierung.
SYS_DATE_PHYSICAL	Ermittelt das jeweils aktuelle Tagesdatum.

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.20 SYS_TIME

Script-Funktion: Stellt die aktuelle Tageszeit zu Beginn der Script-Prozessierung fest.

Syntax

SYS_TIME(*[Zeitformat][,Zeitzone]*)

Syntaxteil	Beschreibung/Format

Zeitformat	Formatvorgabe für die ermittelte Zeit. Format: Script-Literal oder Script-Variable Standard: "HHMMSS"
Zeitzone	Name eines Zeitzonen-Objektes oder Schlüsselwort UTC. Format: Script-Literal oder Script-Variable

Rückgabewert
Aktuelle Uhrzeit im angegebenen Format.

Anmerkungen

Die Script-Funktionen ermittelt die aktuelle Tageszeit. Dabei wird eine Zeitzone, welche als Parameter angegeben wurde, berücksichtigt.

Zeitformat ist optional. Es dient dazu das Format festzulegen, in dem die Funktion ihren Wert zurückgeben soll. Haben Sie *Zeitformat* nicht angegeben, wird das Standardformat "HHMMSS" verwendet.

Zeitzone ist ebenfalls optional. Wird die Script-Funktion ohne diesen Parameter aufgerufen, wird die Zeitzone des Objektes verwendet bzw. die des Mandanten, wenn für das Objekt selbst keine Zeitzone definiert wurde. Wurde eine nicht definierte Zeitzone angegeben, wird automatisch mit dem Standardwert (Zeitzone des Mandanten) gerechnet. Anstatt einer Zeitzone kann auch das Schlüsselwort UTC verwendet werden. Die Uhrzeit wird direkt in UTC (Universal Time Coordinated) zurückgegeben.

Die aktuelle Tageszeit wird zu Beginn der Script-Prozessierung ermittelt und "eingefroren". Damit wird die Konsistenz des Scripts sichergestellt. Wenn Sie die Script-Funktion innerhalb eines Scripts mehrfach verwenden, wird also immer dieselbe Tageszeit zurückgegeben. Das gilt insbesondere auch dann, wenn die Scriptbearbeitung durch eine :WAIT-Anweisung für einen festgelegten Zeitraum unterbrochen wird.

Um die ermittelte Tageszeit mit der Script-Anweisung :PUT_VAR an ein Variablen-Objekt vom Typ "Zeitstempel" zu übergeben, müssen die Zeitformate "HHMMSS" (Standard) oder "HH:MM:SS" verwendet werden. Mit dem Speichern in der Variable geht dieses Zeitformat auf der Windows-Plattform verloren. Die Anzeige der Zeit basiert nun auf den Ländereinstellungen in der Systemsteuerung von Windows.

Die Zeit kann immer nur gemeinsam mit einem Datum in einer Variablen vom Typ "Zeitstempel" gespeichert werden. Dabei sind ausschließlich folgende Kombinationen von Datums- und Zeitformat erlaubt: "JJMMTT HHMMSS" (Standard), "JJJJ-MM-TT HH:MM:SS" und "JJJJMMTT HHMMSS".

Beispiele

Das erste Beispiel ermittelt die aktuelle Tageszeit und übergibt den Wert an eine Script-Variable. Dabei wird eine Zeitzone berücksichtigt, die für mitteleuropäische Zeit definiert wurde. Das Ergebnis wird im Aktivierungsprotokoll ausgegeben.

```
:SET  &ZEIT# = SYS_TIME("HH:MM:SS","TZ.MEZ")
:PRINT  &ZEIT#
```

Im zweiten Beispiel wird ein Teilbegriff verwendet und mit einer Script-Variablen übergeben. Es werden nur die Sekunden der aktuellen Tageszeit festgestellt.

```
:SET  &FORMAT# = "SS"
:SET  &ZEIT# = SYS_TIME(&FORMAT#)
```

Wird die Funktion ohne die Angabe von *Zeitformat* und *Zeitzone* verwendet, ist die Syntax wie im folgenden Beispiel.

```
:IF  SYS_TIME() = "120000"
!...
:ENDIF
```

Im dritten Beispiel werden das aktuelle Tagesdatum und die aktuelle Tageszeit ermittelt und in einem Variablen-Objekt vom Typ "Zeitstempel" gespeichert. Für Datums- und Zeitformat gelten die Standardwerte.

```
:SET  &DATUM# = SYS_DATE()
:SET  &ZEIT# = SYS_TIME()
:PUT_VAR BUCHUNGSDATUM, , "&DATUM# &ZEIT#"
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_TIME_PHYSICAL	Stellt die jeweils aktuelle Tageszeit fest.
SYS_TIMESTAMP_PHYSICAL	Liefert aktuelles Datum und Uhrzeit.

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.21 SYS_TIME_PHYSICAL

Script-Funktion: Stellt die jeweils aktuelle Tageszeit fest.

Syntax

SYS_TIME_PHYSICAL(*[Zeitformat][,Zeitzone]***)**

Syntaxteil	Beschreibung/Format
Zeitformat	Formatvorgabe für die ermittelte Zeit. Format: Script-Literal oder Script-Variable Standard: "HHMMSS"
Zeitzone	Name eines Zeitzonen-Objektes oder Schlüsselwort UTC. Format: Script-Literal oder Script-Variable

Rückgabewert
Aktuelle Uhrzeit im angegebenen Format.

Anmerkungen

Die Script-Funktionen ermittelt die jeweils aktuelle Tageszeit. Dabei wird eine Zeitzone, welche als Parameter angegeben wurde, berücksichtigt.

⚠ Wird die Script-Funktion mehrmals in einem Script verwendet, kann das unter Umständen zu unterschiedlichen Ergebnissen führen (zum Beispiel bei sekunden- oder minutengenauer Abfrage). Damit unterscheidet sich die Script-Funktion von SYS_TIME, welche die aktuelle Tageszeit zu Beginn der Script-Prozessierung ermittelt und dann "einfriert", um die Konsistenz des Scripts zu gewährleisten.

Zeitformat ist optional. Es dient dazu, das Format festzulegen, in dem die Funktion ihren Wert zurückgeben soll. Haben Sie *Zeitformat* nicht angegeben, wird das Standardformat "HHMMSS" verwendet.

Zeitzone ist ebenfalls optional. Wird die Script-Funktion ohne diesen Parameter aufgerufen, wird die Zeitzone des Objektes verwendet bzw. die des Mandanten, wenn für das Objekt selbst keine Zeitzone definiert wurde. Wurde eine nicht definierte Zeitzone angegeben, wird automatisch mit dem Standardwert (Zeitzone des Mandanten) gerechnet. Anstatt einer Zeitzone kann auch das Schlüsselwort UTC verwendet werden. Die Uhrzeit wird direkt in UTC (Universal Time Coordinated) zurückgegeben.

Beispiele

Das erste Beispiel ermittelt jeweils aktuelle Tageszeiten und übergibt die Werte an Script-Variablen. Das Ergebnis sind zwei Zeiten, die mindestens 10 Sekunden auseinander liegen.

```
:SET  &ZEIT1# = SYS_TIME_PHYSICAL("HH:MM:SS")
:WAIT 10
:SET  &ZEIT2# = SYS_TIME_PHYSICAL("HH:MM:SS")
```

Im zweiten Beispiel wird zunächst die aktuelle Tageszeit in UTC ermittelt. Beim zweiten Aufruf der Script-Funktion wird eine Zeitzone angegeben, die für mitteleuropäische Zeit definiert wurde. Das Ergebnis sind zwei Zeiten, die mindestens 1 Stunde und 10 Sekunden auseinander liegen.

```
:SET  &ZEIT1# = SYS_TIME_PHYSICAL("HH:MM:SS","UTC")
:WAIT 10
:SET  &ZEIT2# = SYS_TIME_PHYSICAL("HH:MM:SS","TZ.MEZ")
```

Im dritten Beispiel wird ein Teilbegriff verwendet und mit einer Script-Variablen übergeben. Es werden nur die Minuten der aktuellen Tageszeit festgestellt.

```
:SET  &FORMAT# = "MM"
:SET  &ZEIT# = SYS_TIME_PHYSICAL(&FORMAT#)
```

Wird die Funktion ohne die Angabe von *Zeitformat* verwendet, ist die Syntax wie im folgenden Beispiel.

```
:IF  SYS_TIME_PHYSICAL() = "120000"
!...
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung

SYS_TIME	Stellt die aktuelle Tageszeit zu Beginn der Script-Prozessierung fest.
SYS_TIMESTAMP_PHYSICAL	Liefert aktuelles Datum und Uhrzeit.

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.22 SYS_TIMESTAMP_PHYSICAL

Script-Funktion: Liefert aktuelles Datum und Uhrzeit.

Syntax

SYS_TIMESTAMP_PHYSICAL(*[Zeitzone]***)**

Syntaxteil	Beschreibung/Format
Zeitzone	Name eines Zeitzonen-Objektes oder Schlüsselwort UTC. Format: Script-Literal oder Script-Variable Standardwert: "UTC"

Rückgabewert
Aktuelles Datum und Uhrzeit im Format "JJJJ-MM-TT HH:MM:SS"

Anmerkungen

Wird das Schlüsselwort UTC verwendet oder die Script-Funktion ohne Parameter aufgerufen, werden Datum und Uhrzeit in UTC (Coordinated Universal Time) zurückgegeben.

Beispiel

Das Beispiel ermittelt das aktuelle Datum und die aktuelle Uhrzeit in Sydney. Als Parameter wird der Name eines Zeitzonen-Objektes angegeben, welches für die australische Eastern Standard Time definiert wurde. Der Rückgabewert der Script-Funktion wird im Aktivierungsprotokoll ausgegeben.

```
:SET &JETZT# = SYS_TIMESTAMP_PHYSICAL("TZ.SYD")
 :PRINT &JETZT#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
SYS_TIME	Stellt die aktuelle Tageszeit zu Beginn der Script-Prozessierung fest.

SYS_TIME_PHYSICAL	Stellt die jeweils aktuelle Tageszeit fest.

Script-Sprachmittel - Datum und Zeit
Zeit
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.23 VALID_CALE

Script-Funktion: Prüft, ob ein Datum in einem Kalenderbegriff enthalten ist.

Syntax

VALID_CALE*(Datum, Kalender, Kalenderbegriff)*

Syntaxteil	Beschreibung/Format
Datum	Angabe eines Datums im Format "JJMMTT" oder "JJJJMMTT". Format: Script-Literal oder Script-Variable Es ist möglich, das Datum in einem anderen Datumsformat anzugeben. Geben Sie dazu zuerst das gewünschte Datumsformat, ein Trennzeichen (: oder ;) und danach das Datum an. Die Angabe des Datumsformats ist optional.
Kalender	Name des Kalender-Objektes Format: AE-name, Script-Literal oder Script-Variable
Kalenderbegriff	Name des Kalenderbegriffes Format: AE-Name, Script-Literal oder Script-Variable

Rückgabewerte
"Y" - Das Datum ist im Kalenderbegriff enthalten. "N" - Das Datum ist nicht im Kalenderbegriff enthalten.

Anmerkungen

Die Angabe des Datumsformats ist optional. Wird kein bestimmtes Datumsformat verwendet, muss das Datum in den Formaten "JJMMTT" oder "JJJJMMTT" angegeben werden. Als Trennzeichen zwischen Datumsformat und *Datum* sind Doppelpunkt oder Semikolon erlaubt.

Beispiel

Das Beispiel überprüft, ob das aktuelle Tagesdatum innerhalb des Bereitschaftskalenders gültig ist. Der Rückgabewert der Script-Funktion wird an eine Script-Variable übergeben.

```
:SET  &AKTDAT# = SYS_DATE()
:SET &OPERATOR# = VALID_CALE(&AKTDAT#,"Bereitschaft","Arbeitstag")
```

Siehe auch:

Script-Sprachmittel	Beschreibung
VALID_DATE	Prüft, ob ein Datum gültig ist.
VALID_TIME	Prüft, ob eine Zeit gültig ist.

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.24 VALID_DATE

Script-Funktion: Prüft, ob ein Datum gültig ist.

Syntax

VALID_DATE*(Datum)*

Syntaxteil	Beschreibung/Format
Datum	Angabe eines Datums im Format "JJMMTT" oder "JJJJMMTT". Format: Script-Literal oder Script-Variable Es ist möglich, das Datum in einem anderen Datumsformat anzugeben. Geben Sie dazu zuerst das gewünschte Datumsformat, ein Trennzeichen (: oder ;) und danach das Datum an. Die Angabe des Datumsformats ist optional.

Rückgabewerte
"Y" - Es ist ein gültiges Datum. "N" - Es ist kein gültiges Datum.

Anmerkungen

Wird kein bestimmtes Datumsformat verwendet, muss das Datum in den Formaten "JJMMTT" oder "JJJJMMTT" angegeben werden. Als Trennzeichen zwischen Datumsformat und *Datum* sind Doppelpunkt oder Semikolon erlaubt.

Liefert die Funktion den Rückgabewert "N", dann können Sie mit Hilfe der Script-Sprachmittel zur Fehlerbehandlung den aufgetretenen Fehler analysieren.

Beispiel

Das Beispiel überprüft, ob das Jahr 2001 ein Schaltjahr ist. Das Ergebnis (N) wird im Aktivierungsprotokoll ausgegeben.

```
:SET &RET#=VALID_DATE("TT.MM.JJJJ:29.02.2001")
 :PRINT &RET#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
VALID_CALE	Prüft, ob ein Datum in einem Kalenderbegriff enthalten ist.
VALID_TIME	Prüft, ob eine Zeit gültig ist.

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.25 VALID_TIME

Script-Funktion: Prüft, ob eine Zeit gültig ist.

Syntax

VALID_TIME*(Zeit)*

Syntaxteil	Beschreibung/Format
Zeit	Angabe einer Zeit im Format "HHMMSS". Format: Script-Literal oder Script-Variable Es ist möglich, die Zeit in einem anderen Zeitformat anzugeben. Geben Sie dazu zuerst das gewünschte Zeitformat, ein Trennzeichen (;) und danach die Zeit an. Die Angabe des Zeitformats ist optional.

Rückgabewerte
"Y" - Es ist eine gültige Uhrzeit. "N" - Es ist keine gültige Uhrzeit.

Anmerkungen

Wird kein bestimmtes Zeitformat verwendet, muss die Zeit im Standardformat "HHMMSS" angegeben werden. Als Trennzeichen zwischen Zeitformat und *Zeit* ist nur Semikolon erlaubt.

Beispiele

Das Beispiel überprüft, ob die angegebene Zeit gültig ist. Das Ergebnis (N) wird im Aktivierungsprotokoll ausgegeben.

```
:SET &RET#=VALID_TIME("HH:MM:SS;24:00:00")
 :PRINT &RET#
```

Das zweite Beispiel führt zu einem positiven Ergebnis (Y).

```
:SET &RET#=VALID_TIME("HH:MM:SS;00:00:00")
 :PRINT &RET#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
VALID_CALE	Prüft, ob ein Datum in einem Kalenderbegriff enthalten ist.
VALID_DATE	Prüft, ob ein Datum gültig ist.

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.26 WEEK_NR

Script-Funktion: Liefert die zu einem Datum gehörende Kalenderwoche.

Syntax

WEEK_NR*(Datum)*

Syntaxteil	Beschreibung/Format
Datum	Angabe eines Datums im Format "JJMMTT" oder "JJJJMMTT". Format: Script-Literal oder Script-Variable Es ist möglich, das Datum in einem anderen Datumsformat anzugeben. Geben Sie dazu zuerst das gewünschte Datumsformat, ein Trennzeichen (: oder ;) und danach das Datum an. Die Angabe des Datumsformats ist optional.

Rückgabewert
Kalenderwoche (3-stellig)

Anmerkungen

Die Angabe des Datumsformats ist optional. Wird kein bestimmtes Datumsformat verwendet, muss das Datum in den Formaten "JJMMTT" oder "JJJJMMTT" angegeben werden. Als Trennzeichen zwischen Datumsformat und *Datum* sind Doppelpunkt oder Semikolon erlaubt.

⚠ Die Berechnung der Kalenderwoche hängt von der Definition der ersten Woche des Jahres ab! Welche Woche als erste Kalenderwoche gilt, wird vom Administrator in der Variablen UC_CLIENT_SETTINGS mit den Keys "FIRST_WEEK_METHOD" und "FIRST_DAY_OF_WEEK" festgelegt.

Beispiel

Das Beispiel ermittelt die Kalenderwoche "052" für den angegebenen Tag.

```
:SET &KW# = WEEK_NR("991231")
```

Siehe auch:

Script-Sprachmittel	Beschreibung
WEEKDAY_NR	Liefern den Wochentag eines Datums als Ziffer.
WEEKDAY_XX	Liefert den Wochentag eines Datums als Kürzel.

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.27 WEEKDAY_NR

Script-Funktionen: Liefern den Wochentag eines Datums als Ziffer.

Syntax

WEEKDAY_NR*(Datum)*

Syntaxteil	Beschreibung/Format
Datum	Angabe eines Datums im Format "JJMMTT" oder "JJJJMMTT". Format: Script-Literal oder Script-Variable Es ist möglich, das Datum in einem anderen Datumsformat anzugeben. Geben Sie dazu zuerst das gewünschte Datumsformat, ein Trennzeichen (: oder ;) und danach das Datum an. Die Angabe des Datumsformats ist optional.

Rückgabewerte
"1" - Montag "2" - Dienstag "3" - Mittwoch "4" - Donnerstag "5" - Freitag "6" - Samstag "7" - Sonntag

Anmerkungen

Die Angabe eines Datumsformats ist optional. Wird kein bestimmtes Datumsformat verwendet, muss das Datum in den Formaten "JJMMTT" oder "JJJJMMTT" angegeben werden. Als

Trennzeichen zwischen Datumsformat und *Datum* sind Doppelpunkt oder Semikolon erlaubt.

Beispiele

Diese Beispiele geben die Werte "5" und "6" für das angegebene Datum zurück.

```
:SET &SYLVESTER_2000# = WEEKDAY_NR('TT.MM.JJ:31.12.99')
 :SET &NEUJAHR_2000# = WEEKDAY_NR("TT.MM.JJJJ:01.01.2000")
```

Siehe auch:

Script-Sprachmittel	Beschreibung
WEEK_NR	Liefert die zu einem Datum gehörende Kalenderwoche.
WEEKDAY_XX	Liefert den Wochentag eines Datums als Kürzel.

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.28 WEEKDAY_XX

Script-Funktion: Liefert den Wochentag eines Datums als Kürzel.

Syntax

WEEKDAY_XX*(Datum)*

Syntaxteil	Beschreibung/Format
Datum	Angabe eines Datums im Format "JJMMTT" oder "JJJJMMTT". Format: Script-Literal oder Script-Variable Es ist möglich, das Datum in einem anderen Datumsformat anzugeben. Geben Sie dazu zuerst das gewünschte Datumsformat, ein Trennzeichen (: oder ;) und danach das Datum an. Die Angabe des Datumsformats ist optional.

Rückgabewerte
"MO" - Montag "DI" - Dienstag "MI" - Mittwoch "DO" - Donnerstag "FR" - Freitag "SA" - Samstag "SO" - Sonntag

Anmerkungen

Die Angabe eines Datumsformats ist optional. Wird kein bestimmtes Datumsformat verwendet, muss das Datum in den Formaten "JJMMTT" oder "JJJJMMTT" angegeben werden. Als Trennzeichen zwischen Datumsformat und *Datum* sind Doppelpunkt oder Semikolon erlaubt.

Beispiele

Diese Beispiele geben die Werte "FR" und "SA" für das angegebene Datum zurück.

```
:SET &SYLVESTER_2000# = WEEKDAY_XX("991231")
:SET &NEUJAHR_2000# = WEEKDAY_XX("TT-MM-JJJJ:01-01-2000")
```

Siehe auch:

Script-Sprachmittel	Beschreibung
WEEK_NR	Liefert die zu einem Datum gehörende Kalenderwoche.
WEEKDAY_NR	Liefern den Wochentag eines Datums als Ziffer.

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.12.29 YEAR_9999

Script-Funktion: Extrahiert das Jahr aus einem angegebenen Datum.

Syntax

YEAR_9999*([Datumsformat:]**Datum**)*

Syntaxteil	Beschreibung/Format
Datumsformat	Formatvorgabe für das angegebene *Datum*.
: oder *;*	Trennzeichen zwischen dem *Datumsformat* und *Datum*.
Datum	Angabe eines Datums, welches dem *Datumsformat* entspricht.
[Datumsformat:] Datum	Format: Script-Literal oder Script-Variable

Rückgabewert
Jahreszahl (4-stellig)

Anmerkungen

Die Angabe von *Datumsformat* ist optional. Wird *Datumsformat* nicht verwendet, muss das Datum in den Formaten "JJMMTT" oder "JJJJMMTT" angegeben werden. Als Trennzeichen zwischen *Datumsformat* und *Datum* sind Doppelpunkt oder Semikolon erlaubt.

Beispiel

Das Beispiel liefert den Wert "2000" als Jahr zum angegebenen Datum.

```
:SET  &JAHR# = YEAR_9999("TT.MM.JJ:31.12.00")
```

Siehe auch:

Script-Sprachmittel - Datum und Zeit
Datums-, Zeit- und Periodenformate
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.13 Rechnen

3.13.1 ADD

Script-Funktion: Führt eine Addition aus.

Syntax

ADD*(Operand1, Operand2)*

Syntaxteil	Beschreibung/Format
Operand1	Erster Rechenausdruck. Format: Script-Literal, Script-Variable oder Zahl ohne Hochkommata
Operand2	Zweiter Rechenausdruck. Format: Script-Literal, Script-Variable oder Zahl ohne Hochkommata

Rückgabewert
Ergebnis der Addition

Anmerkungen

Die Script-Funktion addiert *Operand1* und *Operand2.*

Operand1 und *Operand2* müssen Ausdrücke sein, welche Zahlen im erlaubten Wertebereich derDatentypen entsprechen. Das Ergebnis darf diesen Bereich ebenfalls nicht überschreiten.

Wird das Ergebnis einer Script-Variable zugewiesen, muss diese den entsprechenden Datentyp aufweisen. Beim Versuch eine negative Zahl in eine Variable mit dem Datentyp "unsigned" zu speichern, kommt es zu einem Fehler. Handelt es sich beim Ergebnis um eine Gleitkommazahl und die Zielvariable besitzt den Datentyp "signed" oder "unsigned", so werden die Nachkommastellen abgeschnitten. Der Datentyp "float" unterstützt sowohl negative Zahlen als auch Gleitkommazahlen. Weiters ist auch der Datentyp "string" möglich, wobei das Ergebnis nicht als Zahl, sondern als Zeichenkette abgelegt wird. Der Datentyp von Script-Variablen wird mit dem Sprachmittel :DEFINE festgelegt.

Es zählt immer der Datentyp des Ergebnis, nicht jener der Operanden! Ein negativer und positiver Operand kann eine positive Zahl ergeben, die wiederum in eine Ziel-Variable mit dem Datentyp "unsigned" gespeichert werden kann.

⚠ Beachten Sie, dass es bei Rechenoperationen mit Gleitkommazahlen zu einem ungenauen Ergebnis kommen kann!

ℹ Das Ergebnis wird im 16-stelligen Standardformat zurückgeliefert. Gleitkommazahlen besitzen zusätzlich 16 Nachkommastellen und wenn negative Zahlen unterstützt werden, steht das entsprechende Vorzeichen (+ oder -) an erster Stelle. Um die Formatierung zu ändern verwenden Sie die Script-Funktion FORMAT.

Beispiele

Im ersten Beispiel wird das Ergebnis der Rechenoperation an eine Script-Variable übergeben. Die Angabe von *Operand1* und *Operand2* erfolgt als numerischer Ausdruck.

```
:SET &ERGEBNIS# = ADD(1000,333)
```

In diesem zweiten Beispiel wird die Funktion verwendet, um eine Bedingung zu formulieren.

```
:SET &MAXIMUM# = 3000
:SET &LAUF1# = 5000
:SET &LAUF2# = 2000
!...
:IF ADD(&LAUF1#,&LAUF2#) > &MAXIMUM#
!...
:ENDIF
```

Im dritten Beispiel werden zwei Gleitkommazahlen addiert.

```
:DEFINE &ERGEBNIS#,float
:SET &ERGEBNIS# = ADD(10.31,-5.45)
:P &ERGEBNIS#
```

Das Ergebnis wird wie folgt im Aktivierungsprotokoll ausgegeben:

```
U0020408 +0000000000000004.8600000000000000
```

Eine weitere Möglichkeit der Addition

Im nachstehenden Beispiel wird der Wert einer Script-Variable auf die Summe von 1 plus 1 gesetzt.

```
:SET&ADD# = 1 + 1
:P&ADD#
```

Das Ergebnis wird wie folgt im Aktivierungsprotokoll ausgegeben:

U0020408 0000000000000002

Siehe auch:

Script-Sprachmittel	Beschreibung
SUB	Führt eine Subtraktion aus.
MULT	Führt eine Multiplikation aus.
DIV	Führt eine Division aus.
MOD	Liefert den Restwert einer Division.
RANDOM	Generiert Zufallszahlen.

Script-Sprachmittel - Rechnen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.13.2 DIV

Script-Funktion: Führt eine Division aus.

Syntax

DIV*(Operand1, Operand2)*

Syntaxteil	Beschreibung/Format
Operand1	Erster Rechenausdruck. Format: Script-Literal, Script-Variable oder Zahl ohne Hochkommata
Operand2	Zweiter Rechenausdruck. Format: Script-Literal, Script-Variable oder Zahl ohne Hochkommata

Rückgabewert
Ergebnis der Division

Anmerkungen

Die Script-Funktion dividiert *Operand1* durch *Operand 2*.

Operand1 und *Operand2* müssen Ausdrücke sein, welche Zahlen im erlaubten Wertebereich der Datentypen entsprechen. Das Ergebnis darf diesen Bereich ebenfalls nicht überschreiten.

Der *Operand2* darf nicht Null sein!

Wird das Ergebnis einer Script-Variable zugewiesen, muss diese den entsprechenden Datentyp aufweisen. Beim Versuch eine negative Zahl in eine Variable mit dem Datentyp "unsigned" zu speichern, kommt es zu einem Fehler. Handelt es sich beim Ergebnis um eine Gleitkommazahl und die Zielvariable besitzt den Datentyp "signed" oder "unsigned", so werden die Nachkommastellen abgeschnitten. Der Datentyp "float" hingegen unterstützt sowohl negative Zahlen als auch Gleitkommazahlen. Weiters ist auch der Datentyp "string" möglich, wobei das Ergebnis nicht als

Zahl, sondern als Zeichenkette abgelegt wird. Der Datentyp von Script-Variablen wird bei der Definition mit dem Sprachmittel :DEFINE festgelegt.

Es zählt immer der Datentyp des Ergebnis, nicht jener der Operanden! Beispiel: Zwei negative Operanden ergeben eine positive Zahl, die wiederum in eine Ziel-Variable mit dem Datentyp "unsigned" gespeichert werden kann.

⚠ Beachten Sie, dass es bei Rechenoperationen mit Gleitkommazahlen zu einem ungenauen Ergebnis kommen kann!

Das Ergebnis wird im 16-stelligen Standardformat zurückgeliefert. Gleitkommazahlen besitzen zusätzlich 16 Nachkommastellen und wenn negative Zahlen unterstützt werden, steht das entsprechende Vorzeichen (+ oder -) an erster Stelle. Um die Formatierung zu ändern verwenden Sie die Script-Funktion FORMAT.

Beispiele

Im Beispiel wird das Ergebnis "5" der Rechenoperation an eine Script-Variable übergeben. Als *Operand1* und *Operand2* werden Script-Variablen verwendet.

```
:SET  &OP1# = '100'
:SET  &OP2# = '20'
:SET  &ERGEBNIS# = DIV(&OP1#,&OP2#)
```

Wenn der Datentyp der Zielvariable nicht "float" (oder "string") entspricht, so werden Nachkommastellen abgeschnitten. Im folgenden Beispiel wird daher der Wert "0" in die Variable gespeichert.

```
:DEFINE &ERGEBNIS#, unsigned
:SET  &ERGEBNIS# = DIV(10,30)
```

Folgendes Beispiel zeigt eine Division mit Gleitkommazahlen.

```
:DEFINE &ERGEBNIS#,float
:SET &ERGEBNIS# = DIV(-9,-2.25)
:P &ERGEBNIS#
```

Das Ergebnis wird wie folgt im Aktivierungsprotokoll ausgegeben:

```
U0020408 +0000000000000004.0000000000000000
```

Siehe auch:

Script-Sprachmittel	Beschreibung
ADD	Führt eine Addition aus.
SUB	Führt eine Subtraktion aus.
MULT	Führt eine Multiplikation aus.
MOD	Liefert den Restwert einer Division.
RANDOM	Generiert Zufallszahlen.

Script-Sprachmittel - Rechnen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.13.3 GET_BIT

Script-Funktion: Prüft, ob in einem Bit-Feld ein bestimmtes Bit gesetzt ist.

Syntax

GET_BIT*(Zahl, Bit-Position)*

Syntaxteil	Beschreibung/Format
Zahl	Zahl, welche in ein binäres Format (Bit-Feld) umgewandelt wird. Format: Zahl oder Script-Variable
Bit-Position	Position innerhalb des Bit-Feldes, welche geprüft werden soll. Format: Zahl oder Script-Variable

Rückgabewert
"0" - Bit ist gesetzt. "1" - Bit ist nicht gesetzt.

Anmerkungen

Die Script-Funktion wandelt die Zahl, welche mit dem ersten Parameter übergeben wird, in ein binäres Format um. Ergebnis ist ein sogenanntes Bit-Feld. Anschließend wird geprüft, ob das Bit an der angegebenen Position gesetzt ist oder nicht.

⚠ Die *Bit-Position* wird immer von rechts gezählt.

Die Script-Funktion wird auch dazu verwendet, um die 16 Bit-Felder MSG_DESCRIPTOR, MSG_LEVEL und MSG_MISC einer Konsolen-Meldung in z/OS abfragen zu können. Diese können aber auch direkt mit der Script-Funktion GET_CONSOLE abgefragt werden.

Beispiele

Im ersten Beispiel wird geprüft, ob das 3. Bit des Bit-Feldes ("110"), welches der Zahl "6" entspricht, gesetzt ist. Der Rückgabewert "1" (Bit ist gesetzt) wird im Aktivierungsprotokoll ausgegeben.

```
:SET  &RET# = GET_BIT(6, 3)
:PRINT  &RET#
```

Das zweite Beispiel ermittelt einen Bestandteil der Konsolen-Meldung in z/OS. Die zurückgegebene Zahl wird anschließend binär umgerechnet und auf Bit-Position 3 geprüft.

```
:SET  &RET# = GET_CONSOLE("MSG_DESCRIPTOR")
:SET  &RET# = GET_BIT(&RET#, 3)
```

Siehe auch:

Script-Sprachmittel	Beschreibung
GET_CONSOLE	Liest Meldungsdaten des eingetretenen Konsolen-Ereignisses aus.

Script-Sprachmittel - Rechnen

Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.13.4 MOD

Script-Funktion: Liefert den Restwert einer Division.

Syntax

MOD*(Operand1, Operand2)*

Syntaxteil	Beschreibung/Format
Operand1	Erster Rechenausdruck. Format: Script-Literal, Script-Variable oder Zahl ohne Hochkommata
Operand2	Zweiter Rechenausdruck. Format: Script-Literal, Script-Variable oder Zahl ohne Hochkommata

Rückgabewert
Restwert der Division

Anmerkungen

Die Script-Funktion liefert den Rest der Division von *Operand1* durch *Operand2*.

Bei *Operand1* und *Operand2* muss es sich um Ganzzahlen im erlaubten Wertebereich derDatentypen "unsigned" bzw. "signed" handeln! Gleitkommazahlen werden nicht unterstützt! Beim Ergebnis handelt es sich ebenfalls um eine positive oder negative Ganzzahl.

Der *Operand2* darf nicht Null sein!

Beispiel

Im folgenden Beispiel liefert die Funktion das Ergebnis "1".

```
:SET &rest# = MOD(10,3)
```

Das zweite Beispiel hat das Ergebnis "10".

```
:SET &rest# = MOD(10,44)
```

Siehe auch:

Script-Sprachmittel	Beschreibung
ADD	Führt eine Addition aus.
SUB	Führt eine Subtraktion aus.
MULT	Führt eine Multiplikation aus.

DIV	Führt eine Division aus.
RANDOM	Generiert Zufallszahlen.

Script-Sprachmittel - Rechnen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.13.5 MULT

Script-Funktion: Führt eine Multiplikation aus.

Syntax

MULT*(Operand1, Operand2)*

Syntaxteil	Beschreibung/Format
Operand1	Erster Rechenausdruck. Format: Script-Literal, Script-Variable oder Zahl ohne Hochkommata
Operand2	Zweiter Rechenausdruck. Format: Script-Literal, Script-Variable oder Zahl ohne Hochkommata

Rückgabewert
Ergebnis der Multiplikation

Anmerkungen

Die Script-Funktion multipliziert *Operand1* mit *Operand2.*

Operand1 und *Operand2* müssen Ausdrücke sein, welche Zahlen im erlaubten Wertebereich derDatentypen entsprechen. Das Ergebnis darf diesen Bereich ebenfalls nicht überschreiten.

Wird das Ergebnis einer Script-Variable zugewiesen, muss diese den entsprechenden Datentyp aufweisen. Beim Versuch eine negative Zahl in eine Variable mit dem Datentyp "unsigned" zu speichern, kommt es zu einem Fehler. Handelt es sich beim Ergebnis um eine Gleitkommazahl und die Zielvariable besitzt den Datentyp "signed" oder "unsigned", so werden die Nachkommastellen abgeschnitten. Der Datentyp "float" unterstützt sowohl negative Zahlen als auch Gleitkommazahlen. Weiters ist auch der Datentyp "string" möglich, wobei das Ergebnis nicht als Zahl, sondern als Zeichenkette abgelegt wird. Der Datentyp von Script-Variablen wird bei der Definition mit dem Sprachmittel :DEFINE festgelegt.

Es zählt immer der Datentyp des Ergebnis, nicht jener der Operanden! Beispiel: Zwei negative Operanden ergeben eine positive Zahl, die wiederum in eine Ziel-Variable mit dem Datentyp "unsigned" gespeichert werden kann.

⚠ Beachten Sie, dass es bei Rechenoperationen mit Gleitkommazahlen zu einem ungenauen Ergebnis kommen kann!

Das Ergebnis wird im 16-stelligen Standardformat zurückgeliefert. Gleitkommazahlen besitzen zusätzlich 16 Nachkommastellen und wenn negative Zahlen unterstützt werden, steht das entsprechende Vorzeichen (+ oder -) an erster Stelle. Um die Formatierung zu ändern verwenden Sie die Script-Funktion FORMAT.

Beispiel

Im Beispiel wird das Ergebnis der Rechenoperation an eine Script-Variable übergeben. Das Beispiel liefert als Ergebnis "100".

```
:SET  &OP1# = 4
:SET  &OP2# = 25
:SET  &ERGEBNIS# = MULT(&OP1#,&OP2#)
```

Folgendes Beispiel zeigt eine Multiplikation mit Gleitkommazahlen.

```
:DEFINE &ERGEBNIS#,float
:SET &ERGEBNIS# = MULT(-10.31,5.45)
:P &ERGEBNIS#
```

Das Ergebnis wird wie folgt im Aktivierungsprotokoll ausgegeben:

```
U0020408 -0000000000000056.1895000000000000
```

Siehe auch:

Script-Sprachmittel	Beschreibung
ADD	Führt eine Addition aus.
SUB	Führt eine Subtraktion aus.
MOD	Liefert den Restwert einer Division.
DIV	Führt eine Division aus.
RANDOM	Generiert Zufallszahlen.

Script-Sprachmittel - Rechnen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.13.6 RANDOM

Script-Funktion: Generiert Zufallszahlen.

Syntax

RANDOM*(Minimum, Maximum*[, *Basis*]*)*

Syntaxteil	Beschreibung/Format
Minimum	Minimalwert der generierten Zufallszahl. Format: Zahl oder Script-Variable

Maximum	Maximalwert der generierten Zufallszahl. Format: Zahl oder Script-Variable
Basis	Initialwert für die Generierung. Format: Zahl oder Script-Variable

Rückgabewert
Zufällige Zahl im angegebenen Bereich.

Anmerkungen

Die Script-Funktion RANDOM generiert Zufallszahlen. Ein Zahlengenerator erzeugt eine Zahlenreihe, die vom Initialwert - der *Basis* - bestimmt wird.

Wird *Basis* nicht angegeben, ist der Rückgabewert eine sich ständig ändernde Zahl im Bereich zwischen *Minimum* und *Maximum*.

Wird *Basis* angegeben, erfolgt mit diesem Wert eine Initialisierung des Zahlengenerators. Das Ergebnis ist eine Zahlenreihe. Deren Zahlen können durch erneute Aufrufe der Script-Funktion ohne *Basis* gelesen werden. Die generierte Zahlenreihe ist für einen bestimmten Initialwert immer identisch: gleiche Zahlen in gleicher Reihenfolge.

⚠ Wird die Script-Funktion gleichzeitig im Script eines anderen Objektes ausgeführt, überschreibt sie die bereits generierte Zahlenreihe.

Der Script-Funktion können nur positive Ganzzahlen (Datentyp: unsigned) übergeben werden! Das Ergebnis stellt ebenfalls immer eine positive Ganzzahl dar!

Beispiele

- Im ersten Beispiel liefert die Funktion eine Zahl zwischen 1 und 10 (die beiden Zahlen miteingeschlossen). Das Ergebnis lautet z.B. "3".
```
:SET  &zahl# = RANDOM(1, 10)
```
- Im zweiten Beispiel wird zuerst eine Zahlenreihe unter Verwendung des Parameters Basis generiert. Bei weiteren Aufrufen durch die Funktion kann auf die Werte zugegriffen werden. Die ersten 10 Werte der generierten Zahlenreihe mit Basis "1" sind: **6**, 2, 9, 6, 5, 4, 9, 9, 8 und 2. Das Ergebnis, das beim **ersten** Aufruf in "&zahl#" gespeichert wird, ist somit immer "**6**".
```
:SET  &ret# = RANDOM(&min#, &max#, 1)
:SET  &zahl# = RANDOM(&min#, &max#)
```

Bitte beachten Sie, dass es sich bei der genannten Zahlenreihe unter Verwendung von *Basis* "1" um ein Beispiel handelt. Je nach Plattform, auf der RANDOM ausgeführt wird, kann die tatsächlich generierte Zahlenreihe anders aussehen.

Siehe auch:

Script-Sprachmittel	Beschreibung
ADD	Führt eine Addition aus.
SUB	Führt eine Subtraktion aus.
MULT	Führt eine Multiplikation aus.

MOD	Liefert den Restwert einer Division.
DIV	Führt eine Division aus.

Script-Sprachmittel - Rechnen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.13.7 SUB

Script-Funktion: Führt eine Subtraktion aus.

Syntax

SUB*(Operand1, Operand2)*

Syntaxteil	Beschreibung/Format
Operand1	Erster Rechenausdruck. Format: Script-Literal, Script-Variable oder Zahl ohne Hochkommata
Operand2	Zweiter Rechenausdruck. Format: Script-Literal, Script-Variable oder Zahl ohne Hochkommata

Rückgabewert
Ergebnis der Subtraktion

Anmerkungen

Die Script-Funktion subtrahiert *Operand2* von *Operand1*.

Operand1 und *Operand2* müssen Ausdrücke sein, welche Zahlen im erlaubten Wertebereich derDatentypen entsprechen.

Wird das Ergebnis einer Script-Variable zugewiesen, muss diese den entsprechenden Datentyp aufweisen. Beim Versuch eine negative Zahl in eine Variable mit dem Datentyp "unsigned" zu speichern, kommt es zu einem Fehler. Handelt es sich beim Ergebnis um eine Gleitkommazahl und die Zielvariable besitzt den Datentyp "signed" oder "unsigned", so werden die Nachkommastellen abgeschnitten. Der Datentyp "float" unterstützt sowohl negative Zahlen als auch Gleitkommazahlen. Weiters ist auch der Datentyp "string" möglich, wobei das Ergebnis nicht als Zahl, sondern als Zeichenkette abgelegt wird. Der Datentyp von Script-Variablen wird bei der Definition mit dem Sprachmittel :DEFINE festgelegt.

⚠ Beachten Sie, dass es bei Rechenoperationen mit Gleitkommazahlen zu einem ungenauen Ergebnis kommen kann!

Es zählt immer der Datentyp des Ergebnis, nicht jener der Operanden! Ist beispielsweise *Operand2* negativ kann sich eine positive Zahl ergeben, die wiederum in eine Ziel-Variable mit dem Datentyp "unsigned" gespeichert werden kann.

Das Ergebnis wird im 16-stelligen Standardformat zurückgeliefert. Gleitkommazahlen besitzen zusätzlich 16 Nachkommastellen und wenn negative Zahlen unterstützt werden, steht das entsprechende Vorzeichen (+ oder -) an erster Stelle. Um die Formatierung zu ändern verwenden Sie die Script-Funktion FORMAT.

Beispiel

Im Beispiel wird das Ergebnis der Rechenoperation an eine Script-Variable übergeben. Die Angabe von *Operand1* und *Operand2* erfolgt als numerischer Ausdruck.

```
:SET  &ERGEBNIS# = SUB(1000,999)
```

Folgende Beispiel zeigt eine Subtraktion mit Gleitkommazahlen.

```
:DEFINE &ERGEBNIS#,float
:SET &ERGEBNIS# = SUB(10.31,-5.45)
:P &ERGEBNIS#
```

Das Ergebnis wird wie folgt im Aktivierungsprotokoll ausgegeben:

```
U0020408 +0000000000000015.7600000000000000
```

Siehe auch:

Script-Sprachmittel	Beschreibung
ADD	Führt eine Addition aus.
MULT	Führt eine Multiplikation aus.
MOD	Liefert den Restwert einer Division.
DIV	Führt eine Division aus.
RANDOM	Generiert Zufallszahlen.

Script-Sprachmittel - Rechnen
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14 Strings

3.14.1 ARRAY_2_STRING

Script-Funktion: Wandelt ein Script-Array in eine Zeichenkette um.

Syntax

ARRAY_2_STRING*(Script-Array *[, Trennzeichen [, Bereich]]**)***

Syntaxteil	Beschreibung/Format

Script-Array	Variablenname des Arrays Format: Script-Variable
Trennzeichen	Ein oder mehrere Zeichen, die in der Zeichenkette zwischen den Array-Elementen eingefügt werden sollen. Format: Script-Variable oder Script-Literal
Bereich	Bereich des Arrays Erlaubte Werte: "ALL" oder "FILLED" (Standardwert) "ALL" = Alle Elemente des Arrays verwenden "FILLED" = Alle Elemente bis zum letzten befüllten Element speichern.

Rückgabewert
Zeichenkette, welche die Elemente des Arrays enthält.

Anmerkungen

Die Script-Funktion liefert die Elemente eines Script-Arrays in Form einer Zeichenkette zurück. Dabei ist der Variablenname des Arrays, inklusive der leeren Indexklammern [], anzugeben.

Zusätzlich können ein oder mehrere *Trennzeichen* festgelegt werden, welche die Elemente in der resultierenden Zeichenkette voneinander trennen. Wird der Parameter *Trennzeichen* nicht angegeben, ...

Sie haben außerdem die Möglichkeit zu bestimmen, ob alle Elemente oder nur jene bis zum letzten befüllten Element verwendet werden sollen. Im letzteren Fall werden leere Elemente am Ende des Arrays ignoriert.

Beispiel

In folgendem Beispiel-Script wird ein Array erstellt und mit Einträgen eines Variablen-Objektes befüllt. Anschließend wird das Array in eine Zeichenkette umgewandelt und diese im Aktivierungsreport ausgegeben. Als Trennzeichen für die Array-Elemente wird ein Bindestrich verwendet.

```
DEFINE &ARRAY#, string, 5
:DEFINE &STR#, string
:FILL &ARRAY#[] = GET_VAR(VARA.STATIC.TEST,"KEY01")
:SET &STR# = ARRAY_2_STRING(&ARRAY#[],"-",FILLED)
:P "&STR#"
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:CLEAR	Setzt ein Script-Array wieder auf seine Initialwerte zurück.
:DEFINE	Erstellung einer Script-Variable mit einem bestimmten Datentyp.

Script-Sprachmittel - Script-Aufbau und Prozessierung
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.2 ALPHA2RUNNR

Script-Funktion: Konvertiert den Namen einer Job bzw. Reportdatei in eine RunID.

Syntax

ALPHA2RUNNR*(Zeichenfolge)*

Syntaxteil	Beschreibung/Format
Zeichenfolge	7-stellige Zeichenfolge (Buchstaben) Format: Script-Literal oder Script-Variable

Rückgabewert
Laufende Nummer (RunID)

Anmerkungen

Die Script-Funktion konvertiert eine Zeichenfolge, bestehend aus 7 Buchstaben, in eine RunID.

Werden Jobs und Job-Reports im Dateisystem gespeichert, enthalten ihre Dateinamen die laufende Nummer (RunID) des Jobs in Form einer 7-stelligen Zeichenfolge (Buchstaben). Beispiel eines Job-Reports unter Windows: OGMITAEV.TXT. "O" kennzeichnet den Job-Report, "GMITAEV" ist die Zeichenfolge der umgesetzten RunID 2000061045. Für eine Verarbeitung der Dateinamen in einem Script, kann die Zeichenfolge mit ALPHA2RUNNR wieder in die 10-stellige RunID konvertiert werden.

Mit der Script-Funktion RUNNR2ALPHA wird die 10-stellige RunID in eine 7-stellige Zeichenfolge (Buchstaben) konvertiert.

Beispiele

Im Beispiel wird die 7-stellige Zeichenfolge aus dem Dateinamen ermittelt und im Aktivierungsprotokoll ausgegeben.

```
:SET &ALPHA# = MID("JAADMXZT.TXT", 2, 7)
:SET &RET#   = ALPHA2RUNNR(&ALPHA#)
:PRINT "RunID: &RET#"
```

Siehe auch:

Script-Sprachmittel	Beschreibung
RUNNR2ALPHA	Konvertiert RunID in den zugehörigen Dateinamen.

Script-Sprachmittel - Strings
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.3 CONVERT

Script-Funktion: Wandelt den Datentyp eines Wertes um.

Syntax

CONVERT *(Datentyp, Wert)*

Syntaxteil	Beschreibung/Format
Datentyp	Datentyp in welchen umgewandelt werden soll. Erlaubte Werte: "unsigned", "signed", "float" und "string" unsigned: Positive Ganzzahlen ohne Vorzeichen signed: Ganzzahlen mit Vorzeichen float: Gleitkommazahlen string: Zeichenkette, Text Die Angabe von Datentypen erfolgt nicht mit Hochkomma!
Wert	Wert, dessen Datentyp in einen anderen Datentyp umgewandelt werden soll. Format: Script-Literal oder Script-Variable Zahlen müssen auch unter Hochkomma angegeben werden!

Rückgabewert
Wert mit dem umgewandelten Datentyp.

Anmerkungen

Mit der Script-Funktion ist es möglich den Datentyp eines Wertes, der entweder direkt oder über eine Script-Variable angeben wird, in einen anderen Datentyp umzuwandeln. Der Rückgabewert ist der umgewandeltete Wert, der einer Zielvariable zugewiesen werden muss.

Beachten Sie bei der Verwendung dieser Script-Funktion, dass der Datentyp des umzuwandelnden Wertes zum Datentyp der Zielvariable passt.

Strings können nur in Zahlen umgewandelt werden, wenn die Zeichenkette eine Zahl in einem gültigen Format beinhaltet!

Wenn versucht wird, einen höheren Zahlentyp in einen niedrigeren umzuwandeln, werden Kommazahlen gerundet oder die Vorzeichen entfernt.

Entspricht der Datentyp der Zielvariable nicht dem Parameter "Datentyp" der Funktion, ist die Umwandlung nicht möglich und es kommt zu einem Scriptfehler!

⚠ Beachten Sie, dass mit der Scriptfunktion keine negativen Zahlen umgewandelt werden können!

Beispiel

Im ersten Beispiel wird eine positive Ganzzahl in eine Zeichenkette umgewandelt.

```
:define &unsigned#, unsigned
:define &string#, string

:set &unsigned# = 12
:set &string# = CONVERT(string, &unsigned#)
```

Im zweiten Beispiel wird eine Zeichenkette in eine Zahl umgewandelt. Dies ist nur möglich, wenn die Zeichenkette aus einer Zahl besteht, welche ein für den Zieldatentyp gültiges Format aufweist.

```
:define &unsigned#, unsigned
:define &string#, string

:set &string# = "1234"
:set &unsigned# = CONVERT(unsigned, &string#)
```

Siehe auch:

Script-Sprachmittel - Objekte bearbeiten
Import und Export von Objekten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.4 FORMAT

Script-Funktion: Änderung der Formatierung einer Zahl

Syntax

FORMAT*(Zahl *[,Format]**)***

Syntaxteil	Beschreibung/Format
Zahl	Zahl, die formatiert werden soll. Format: Script-Literal oder Script-Variable
Format	Nullen, die als Platzhalter für die Stellen der Zahl fungieren. Bei Gleitkommazahlen ist zusätzlich die Angabe eines Trennzeichens als Dezimalpunkt und die Anzahl der Nachkommastellen möglich. Format: Script-Literal oder Script-Variable Standardwert: 0

Rückgabewert
Formatierte Zahl (Datentyp: string)

Anmerkungen

Mit der Script-Funktion können führende Nullen einer Ganzzahl hinzugefügt oder entfernt werden. Bei Gleitkommazahlen ist es möglich, die Anzahl der Nachkommastellen festzulegen.

Durch die Verwendung des Parameters *Format*, kann die Anzahl der führenden Nullen von Ganzzahlen festgelegt werden. Die Anzahl an Nullen, die Sie in diesem Parameter vermerken, dienen als Platzhalter für die Gesamtanzahl der Stellen. Ist die Anzahl der angegebenen Stellen

kleiner als die der Zahl, so bleibt der Wert unverändert. Geben Sie diesen Parameter nicht an, so entfernt die Funktion ihre führenden Nullen.

Bei Gleitkommazahlen kann zusätzlich die Anzahl der Kommastellen angegeben werden. Geben Sie dazu einen Dezimalpunkt als Trennzeichen beim Parameter *Format* an. Mit den Nullen nach dem Trennzeichen wird die Anzahl der Nachkommastellen definiert.
Nachkommastellen, die über die diese Anzahl hinausgehen, werden abgeschnitten. Es erfolgt keine Rundung! Wird der Dezimalpunkt nicht angegeben, so werden alle Nachkommastellen entfernt! Besitzt die Zahl weniger Kommastellen als bei *Format* angegeben, so wird der Rest mit Nullen aufgefüllt.

Wird zusätzlich bei *Format* ein '+' als erstes Zeichen angegeben (zum Beispiel: "+0.00"), wird das Vorzeichen auch bei positiven Zahlen angezeigt.

Die Zielvariable, die den Rückgabewert der Funktion zugewiesen bekommt, muss den Datentyp "string" aufweisen!

Ist das Ergebnis durch Entfernung aller Nachkommastellen 0, so spielt das Vorzeichen keine Rolle mehr und es wird entfernt.

Beispiele

Das erste Beispiel entfernt die führenden Nullen beim 16-stelligen Rückgabewert einer Script-Funktion. Das Ergebnis wird im Aktivierungsprotokoll ausgegeben.

```
:SET  &SRV#=SYS_BUSY_60
:SET  &RET#=FORMAT(&SRV#)
:PRINT  &RET#
```

Im zweiten Beispiel wird die angegebene Zahl 5-stellig formatiert. Das Ergebnis (00125) wird im Aktivierungsprotokoll ausgegeben.

```
:SET &RET#=FORMAT("125","00000")
:PRINT  &RET#
```

Auch im dritten Beispiel werden die führenden Nullen entfernt, weil die Anzahl der Stellen zu klein angegeben wurde. Das Ergebnis (333) wird im Aktivierungsprotokoll ausgegeben.

```
:SET  &RET#=FORMAT("0000333","00")
:PRINT  &RET#
```

Im vierten Beispiel bleibt die Zahl unverändert.

```
:SET  &RET#=FORMAT("555","00")
:PRINT  &RET#
```

Das fünfte Beispiel zeigt die Formatierung einer Gleitkommazahl auf eine Nachkommastelle. Es wird "-0.7" im Aktivierungsprotokoll ausgegeben.

```
:DEFINE &NUM#,float
:DEFINE &RET#,string
:SET &NUM#=-0.75
:SET  &RET#=FORMAT(&NUM#,"00.0")
:PRINT  &RET#
```

Im sechsten Beispiel werden die Kommastellen entfernt. Die Ausgabe im Aktivierungsreport ist "0000".

```
:DEFINE &NUM#,float
:DEFINE &RET#,string
:SET &NUM#=0.65
```

```
:SET  &RET#=FORMAT(&NUM#,"0000")
:PRINT  &RET#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
STR_LTRIM	Entfernt Leerzeichen am Anfang einer Zeichenkette.
STR_RTRIM	Entfernt Leerzeichen am Ende einer Zeichenkette.

Script-Sprachmittel - Strings
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.5 HEX

Script-Funktion: Konvertiert eine Zeichenkette in hexadezimale Schreibweise.

Syntax

HEX*(Zeichenkette)*

Syntaxteil	Beschreibung/Format
Zeichenkette	Alphanumerische Zeichenfolge, welche konvertiert werden soll. Format: Script-Literal oder Script-Variable

Rückgabewert
Zeichenkette mit maximal 252 Zeichen in hexadezimaler Schreibweise.

Anmerkungen

Mit der Script-Funktion setzen Sie jedes einzelne Zeichen einer *Zeichenkette* entsprechend des hexadezimalen Zeichensatzes um.

Der Script-Funktion wird eine *Zeichenkette* übergeben, welche aus maximal 126 Zeichen bestehen darf. Ist *Zeichenkette* länger, wird nach 126 Zeichen ohne eine Meldung abgeschnitten.

Beispiel

Im Beispiel wird die Zeichenkette in hexadezimale Schreibweise umgesetzt. Das Ergebnis "554334" wird im Aktivierungsprotokoll ausgegeben.

```
:SET &RET#=HEX("automic")
:PRINT &RET#
```

Siehe auch:

Script-Sprachmittel - Strings

Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.6 ISNUMERIC

Script-Funktion: Prüft, ob eine Zeichenkette numerisch ist.

Syntax

ISNUMERIC*(Zeichenkette)*

Syntaxteil	Beschreibung/Format
Zeichenkette	Zeichenfolge, welche geprüft werden soll. Format: Script-Literal oder Script-Variable

Rückgabewerte
"Y" - die Zeichenkette ist numerisch "N" - die Zeichenkette ist nicht numerisch

Anmerkungen

Mit der Script-Funktion prüfen Sie, ob alle Zeichen einer *Zeichenkette* numerisch sind.

Beispiele

Beide Beispiele liefern das Ergebnis "Y".

```
:SET  &RET#=ISNUMERIC("123")
 :PRINT  &RET#

:SET  &RET#=ISNUMERIC("00123")
 :PRINT  &RET#
```

Im folgenden Fall ist der String nicht numerisch. Die Funktion gibt daher den Wert "N" zurück.

```
:SET  &RET#=ISNUMERIC("1abc")
 :PRINT  &RET#
```

Siehe auch:

Script-Sprachmittel - Strings
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.7 RUNNR2ALPHA

Script-Funktion: Konvertiert RunID in den zugehörigen Dateinamen.

Syntax

RUNNR2ALPHA*(RunID)*

Syntaxteil	Beschreibung/Format
RunID	10-stellige laufende Nummer (RunID) Format: Script-Literal, Zahl oder Script-Variable

Rückgabewert
Name der Job- bzw. Reportdatei.

Anmerkungen

Die Script-Funktion konvertiert eine 10-stellige RunID in eine Zeichenfolge, bestehend aus 7 Buchstaben.

Für die Dateinamen von Jobs und Job-Reports, die im Dateisystem gespeichert werden, wird die 10-stellige RunID in eine 7-stelligen Zeichenfolge (Buchstaben) umgesetzt. Beispiel eines Job-Reports unter Windows: OGMITAEVN.TXT. "O" kennzeichnet den Job-Report, "GMITAEVN" ist die Zeichenfolge der umgesetzten RunID 2000061045. Für eine Verarbeitung der Dateinamen in einem Script, kann die gebildete Zeichenfolge (Buchstaben) aus der 10-stelligen RunID ermittelt werden.

Für den eigenen Job liefert die Script-Funktion GET_ATT mit den Attributen FILENAME_JOB bzw. FILENAME_SYSOUT die Dateinamen des Jobs und des Job-Reports.

Mit der Script-Funktion ALPHA2RUNNR wird die 7-stellige Zeichenfolge (Buchstaben) in die 10-stellige RunID konvertiert.

Beispiele

Im Beispiel wird der Job "MAWI.TAG" aktiviert. Die dabei zurückgegebene RunID wird in die 7-stellige Zeichenfolge konvertiert und im Aktivierungsprotokoll ausgegeben.

```
:SET   &RUNNR# = ACTIVATE_UC_OBJECT("MAWI.TAG")
:SET   &RET#   = RUNNR2ALPHA(&RUNNR#)
:PRINT  "ALPHA: &RET#"
```

Siehe auch:

Script-Sprachmittel	Beschreibung
ALPHA2RUNNR	Konvertiert den Namen einer Job bzw. Reportdatei in eine RunID.

Script-Sprachmittel - Strings
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.8 STR_CAT

Script-Funktion: Verbindet zwei Zeichenketten zu einer neuen Zeichenkette.

Syntax

STR_CAT*(Zeichenkette1, Zeichenkette2)*

Syntaxteil	Beschreibung/Format
Zeichenkette1, *Zeichenkette2*	Alphanumerische Zeichenfolge Format: Script-Literal oder Script-Variable

Rückgabewert
Zeichenkette, die aus den beiden angegebenen Zeichenketten besteht

Beispiel

Das Beispiel nutzt die Funktion, um einen Titel (z.B. "Tagesauswertung 12.01.2005") zu erstellen und diesen in einer Script-Variablen zu hinterlegen.

```
:SET  &VORGANG# = "Tagesauswertung "
 :SET  &DATUM# = SYS_DATE("TT.MM.JJJJ")
 :SET  &TITEL# = STR_CAT(&VORGANG#, &DATUM#)
```

Siehe auch:

Script-Sprachmittel - Strings
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.9 STR_CUT, MID, SUBSTR

Script-Funktionen: Kopieren Zeichen aus einer Zeichenkette.

Syntax

STR_CUT*(Zeichenkette, Beginn[, Länge])*
MID*(Zeichenkette, Beginn[, Länge])*
SUBSTR*(Zeichenkette, Beginn[, Länge])*

Syntaxteil	Beschreibung/Format
Zeichenkette	Alphanumerische Zeichenfolge Format: Script-Literal oder Script-Variable

Beginn	Position, ab dem Zeichen kopiert werden sollen Format: Zahl ohne Hochkommata oder Script-Variable
Länge	Anzahl der Zeichen, die kopiert werden sollen Format: Zahl ohne Hochkommata oder Script-Variable

Rückgabewert
Teil einer Zeichenkette

Anmerkungen

Die Script-Funktionen sind funktional identisch. Sie kopieren Zeichen aus einer angegebenen *Zeichenkette*. Die *Länge* ist ein optionaler Parameter. Wird sie nicht angegeben, so werden alle Zeichen bis zum Ende der *Zeichenkette* geliefert.

Die Zeichenkette, aus der kopiert wird, bleibt natürlich unverändert.

Beispiele

Als Wert liefert die Funktion im ersten Beispiel "CD" zurück. Das Ergebnis des zweiten Beispiels ist "CDEFGH", die bis zum Ende kopierte Zeichenfolge.

```
:SET &STRING# = MID("ABCDEFGH",3,2)
```

```
:SET &STRING# = STR_CUT("ABCDEFGH",3)
```

Im folgenden Beispiel wird die Funktion dazu verwendet, um einen vom Benutzer eingegebenen Begriff aufzuteilen. Die ersten drei Zeichen werden dabei der ersten Script-Variablen, das letzte Zeichen der zweiten Script-Variablen zugewiesen.

```
:READ &TABNAME#, "04", "Bitte Tabellennamen xxxy eingeben"
 :SET &TABPRE# = SUBSTR(&TABNAME#,1,3)
 :SET &TABSUF# = SUBSTR(&TABNAME#,4,1)
```

Siehe auch:

Script-Sprachmittel	Beschreibung
STR_SUBSTITUTE	Ersetzt Zeichen oder Zeichenkette innerhalb einer Zeichenkette.
STR_CAT	Verbindet zwei Zeichenketten zu einer neuen Zeichenkette.

Script-Sprachmittel - Strings
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.10 STR_ENDS_WITH

Script-Funktion: Prüft ob eine Zeichenkette mit einer bestimmten anderen Zeichenkette endet.

Syntax

STR_ENDS_WITH*(Zeichenkette1, Zeichenkette2)*

Syntaxteil	Beschreibung/Format
Zeichenkette1	Zeichenfolge, die geprüft werden soll. Format: Script-Literal oder Script-Variable
Zeichenkette2	Zeichenfolge, welche zur Prüfung verwendet wird. Format: Script-Literal oder Script-Variable

Rückgabewert
"Y" = Zeichenkette1 endet mit Zeichenkette2 "N" = Zeichenkette1 schließt nicht mit Zeichenkette2 ab

Anmerkungen

Das Sprachmittel prüft ob das Ende einer bestimmten Zeichenkette einer anderen Zeichenkette gleicht. Anzugeben ist dabei die zu prüfende Zeichenkette (Parameter *Zeichenkette1*) und das Ende, ebenfalls in Form einer Zeichenkette (Parameter *Zeichenkette2*).

Die Script-Funktion liefert auch "Y", wenn *Zeichenkette1* und *Zeichenkette2* ident sind.

Beachten Sie, dass Klein- und Großschreibung beim Vergleich beachtet wird.

Beispiel

Folgendes Beispielscript wird in einem FileTransfer-Objekt verwendet und prüft, ob die Zieldatei den Namen "test.txt" besitzt. Wenn ja, wird eine Meldung im Aktivierungsreport protokolliert.

```
:SET &DST# = GET_ATT(FT_DST_FILE)
:SET &VAR# = STR_LC(&DST#)
:SET &CHECK# = STR_ENDS_WITH(&VAR#,"test.txt")
:IF &CHECK# EQ "Y"
: P "Zieldatei = test.txt"
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
STR_STARTS_WITH	Prüft ob eine Zeichenkette mit einer bestimmten anderen Zeichenkette beginnt.

Script-Sprachmittel - Strings
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.11 STR_FIND

Script-Funktion: Sucht ein Zeichen oder eine Zeichenkette innerhalb einer Zeichenkette.

Syntax

STR_FIND*(Zeichenkette1, Zeichenkette2[, Start])*

Syntaxteil	Beschreibung/Format
Zeichenkette1	Alphanumerische Zeichenfolge in der gesucht werden soll. Format: Script-Literal oder Script-Variable
Zeichenkette2	Einzelnes Zeichen oder alphanumerische Zeichenfolge, die gesucht wird. Format: Script-Literal oder Script-Variable
Start	Position, ab der die Suche beginnen soll. Format: Zahl ohne Hochkommata Standardwert: 1

Rückgabewerte
Position an der das Zeichen oder die Zeichenkette gefunden wurde. "0" - Das Zeichen oder die Zeichenkette wurde nicht gefunden.

Anmerkungen

Die Funktion sucht die *Zeichenkette2* innerhalb der *Zeichenkette1* und beginnt dabei ab *Start* mit der Suche. Wird *Start* nicht angegeben, beginnt die Suche ab Position 1. Die *Zeichenkette2* darf auch aus einem einzigen Zeichen bestehen. Die Suche ist case-insensitiv. Groß- und Kleinschreibung wird daher nicht berücksichtigt.

Die Script-Funktion liefert die erste Position, an der die *Zeichenkette2* gefunden wurde, als Wert zurück. Die Position wird dabei immer vom Anfang der *Zeichenkette1* gezählt, nicht etwa ab *Start*.

Beispiel

Im Beispiel liefert die Suche nach "#" innerhalb der Zeichenkette "AUTOMIC#01, AUTOMIC#02" den Wert 4 zurück. Er wird im Aktivierungsprotokoll ausgegeben.

```
:SET &STRING#="AUTOMIC#01, AUTOMIC#02"
:SET &SUCHSTRING#="#"
:SET &POS#=STR_FIND(&STRING#,&SUCHSTRING#)
:PRINT &POS#
```

Im folgenden Beispiel beginnt die Suche erst ab der 2. Stelle. Die Position an der die Zeichenkette gefunden wird ist somit 9.

```
:SET &POS#=STR_FIND("AUTOMIC#01, AUTOMIC#02","AUTOMIC", 2)
:PRINT &POS#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
STR_FIND_REVERSE	Sucht ein Zeichen oder eine Zeichenkette innerhalb einer Zeichenkette. Die Suche beginnt am Ende der zu durchsuchenden Zeichenkette.

Script-Sprachmittel - Strings
Beispielsammlung
Benachrichtigung mit variablem Meldungstext
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.12 STR_FIND_REVERSE

Script-Funktion: Sucht ein Zeichen oder eine Zeichenkette innerhalb einer Zeichenkette. Die Suche beginnt am Ende der zu durchsuchenden Zeichenkette.

Syntax

STR_FIND_REV[ERSE]*(Zeichenkette1, Zeichenkette2)*

Syntaxteil	Beschreibung/Format
Zeichenkette1	Alphanumerische Zeichenfolge in der gesucht werden soll. Format: Script-Literal oder Script-Variable
Zeichenkette2	Einzelnes Zeichen oder alphanumerische Zeichenfolge, die gesucht wird. Format: Script-Literal oder Script-Variable

Rückgabewerte
Position an der das Zeichen oder die Zeichenkette gefunden wurde. "0" - Das Zeichen oder die Zeichenkette wurde nicht gefunden.

Anmerkungen

Die Script-Funktion sucht die *Zeichenkette2* innerhalb der *Zeichenkette1*. Im Gegensatz zur Script-Funktion STR_FIND wird dabei die *Zeichenkette1* vom Ende beginnend durchsucht.

Die *Zeichenkette2* darf aus einem oder mehreren Zeichen bestehen. Die Suche ist case-insensitiv. Groß- und Kleinschreibung wird daher nicht berücksichtigt.

Die Script-Funktion liefert die erste Position, an der die *Zeichenkette2* gefunden wurde, als Wert zurück. Die Position wird dabei vom Anfang der *Zeichenkette1* gezählt.

Beispiel

Im Beispiel liefert die Suche nach "#" innerhalb der Zeichenkette "AE#01, AE#02" den Wert 12 zurück. Er wird im Aktivierungsprotokoll ausgegeben.

```
:SET &STRING#="AE#01, AE#02"
:SET &SUCHSTRING#="#"
:SET &POS#=STR_FIND_REVERSE(&STRING#,&SUCHSTRING#)
:PRINT &POS#
```

Im folgenden Beispiel wird eine Zeichenkette gesucht. Die Position an der diese gefunden wird ist 9.

```
:SET &POS#=STR_FIND_REV("AE#01, AE#02","AE")
:PRINT &POS#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
STR_FIND	Sucht ein Zeichen oder eine Zeichenkette innerhalb einer Zeichenkette.

Script-Sprachmittel - Strings
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.13 STR_ISLOWER

Script-Funktion: Prüft die Buchstaben einer Zeichenkette auf Kleinschreibung.

Syntax

STR_ISLOWER*(Zeichenkette)*

Syntaxteil	Beschreibung/Format
Zeichenkette	Alphanumerische Zeichenfolge Format: Script-Literal oder Script-Variable

Rückgabewert
"Y" = Alle Buchstaben der Zeichenkette besitzen Kleinschreibung. "N" = Die Zeichenkette enthält keine Buchstaben oder zumindest einen Großbuchstaben.

Anmerkungen

Die Script-Funktion überprüft, ob alle Buchstaben einer angegebenen Zeichenkette klein geschrieben sind. Sonderzeichen und Zahlen in der Zeichenkette werden dabei ignoriert. Die Zeichenkette kann dabei entweder direkt (unter einfache oder doppelte Hochkommata) oder in Form einer Script-Variable angegeben werden.

Enthält die Zeichenkette mindestens einen Großbuchstaben oder gar keine Buchstaben, so liefert die Script-Funktion "N" zurück. Besitzen alle Buchstaben in der Zeichenkette Kleinschreibung, so liefert das Sprachmittel "Y" zurück.

Beispiel

Folgendes Beispiel prüft, ob der Archive Key des eigenen Objektes nur Kleinbuchstaben besitzt.

```
:SET &VAR# = GET_ATT(ARCHIVE_KEY1)
:SET &CHECK# = STR_ISLOWER(&VAR#)
:IF &CHECK# EQ "Y"
: P "ArchiveKey1 enthält nur Kleinbuchstaben."
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
STR_ISUPPER	Prüft die Buchstaben einer bestimmten Zeichenkette auf Großschreibung.

Script-Sprachmittel - Strings
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.14 STR_ISUPPER

Script-Funktion: Prüft die Buchstaben einer bestimmten Zeichenkette auf Großschreibung.

Syntax

STR_ISUPPER*(Zeichenkette)*

Syntaxteil	Beschreibung/Format
Zeichenkette	Alphanumerische Zeichenfolge Format: Script-Literal oder Script-Variable

Rückgabewert
"Y" = Alle Buchstaben der Zeichenkette besitzen Großschreibung. "N" = Die Zeichenkette enthält keine Buchstaben oder zumindest einen Kleinbuchstaben.

Anmerkungen

Die Script-Funktion überprüft, ob alle Buchstaben einer angegebenen Zeichenkette groß geschrieben sind. Sonderzeichen und Zahlen in der Zeichenkette werden dabei ignoriert. Die Zeichenkette kann dabei entweder direkt (unter einfache oder doppelte Hochkommata) oder in Form einer Script-Variable angegeben werden.

Enthält die Zeichenkette mindestens einen Kleinbuchstaben oder gar keine Buchstaben, so liefert die Script-Funktion "N" zurück. Besitzen alle Buchstaben in der Zeichenkette Großbschreibung, so liefert das Sprachmittel "Y" zurück.

Beispiel

Folgendes Beispiel liest einen Eintrag aus einem VARA-Objekt und prüft diesen auf Großschreibung.

```
:SET &VAR# = GET_VAR(VARA.DB, "TEST", 1)
:SET &CHECK# = STR_ISUPPER(&VAR#)
:IF &CHECK# EQ "Y"
: P "STR_ISUPPER(&VAR#) = Y"
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
STR_ISLOWER	Prüft die Buchstaben einer Zeichenkette auf Kleinschreibung.

Script-Sprachmittel - Strings
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.15 STR_LC, CONV_LC

Script-Funktionen: Setzen alle Großbuchstaben einer Zeichenkette auf Kleinbuchstaben um.

Syntax

STR_LC*(Zeichenkette)*
CONV_LC*(Zeichenkette)*

Syntaxteil	Beschreibung/Format
Zeichenkette	Alphanumerische Zeichenfolge Format: Script-Literal oder Script-Variable

Rückgabewert
Zeichenkette in der alle Buchstaben Kleinbuchstaben sind.

Anmerkungen

Beide Script-Funktionen sind funktional identisch.

Beispiele

Beide Beispiele setzen eine *Zeichenkette* in Kleinbuchstaben um. Die *Zeichenkette* wird der Funktion im ersten Beispiel als String-Literal, im zweiten als Script-Variable übergeben. Die Ergebnisse der Funktion lauten "abcdefgh 123&%$§" sowie "ucaagx.htm".

```
:SET &STRING# = CONV_LC("ABCDEFGH 123&%$§")
```

```
:SET &NAME# = "UCAAGX.HTM"
 :SET &STRING# = STR_LC(&NAME#)
```

Siehe auch:

Script-Sprachmittel	Beschreibung
CONV_UC oder STR_UC	Setzen alle Kleinbuchstaben einer Zeichenkette auf Großbuchstaben um.

Script-Sprachmittel - Strings
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.16 STR_LENGTH, STR_LNG

Script-Funktionen: Ermitteln die Länge einer Zeichenkette.

Syntax

STR_LENGTH*(Zeichenkette)*
STR_LNG*(Zeichenkette)*

Syntaxteil	Beschreibung/Format
Zeichenkette	Alphanumerische Zeichenfolge Format: Script-Literal oder Script-Variable

Rückgabewert
Anzahl der Zeichen einer Zeichenkette

Anmerkungen

Beide Script-Funktionen sind funktional identisch.

Beispiele

Das erste Beispiel liefert 18 als die ermittelte Zeichenanzahl.

```
:SET &anzahl# = STR_LENGTH(" Automic software ")
```

Das zweite Beispiel ermittelt die Länge einer Zeichenkette, die in einer Script-Variable übergeben wird.

```
:SET &anzahl# = STR_LENGTH(&zeichenfolge)
```

Siehe auch:

Script-Sprachmittel - Strings
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.17 STR_LTRIM

Script-Funktion: Entfernt Leerzeichen am Anfang einer Zeichenkette.

Syntax

STR_LTRIM*(Zeichenkette)*

Syntaxteil	Beschreibung/Format
Zeichenkette	Alphanumerische Zeichenfolge, welche bearbeitet werden soll. Format: Script-Literal oder Script-Variable

Rückgabewert
Zeichenkette ohne führende Leerzeichen

Beispiel

Das Beispiel gibt das Ergebnis ("Datenbank wird geöffnet... ") im Aktivierungsprotokoll aus.

```
:SET &RET# = STR_LTRIM("   Datenbank wird geöffnet...   ")
 :PRINT &RET#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
STR_RTRIM	Entfernt Leerzeichen am Ende einer Zeichenkette.
STR_TRIM	Entfernt Leerzeichen am Anfang und Ende einer Zeichenkette.

Script-Sprachmittel - Strings
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.18 STR_MATCH

Script-Funktion: Vergleicht zwei Zeichenketten.

Syntax

STR_MATCH*(Zeichenkette1, Zeichenkette2*[, *Wildcard1*[, *Wildcard2*]]*)*

Syntaxteil	Beschreibung/Format
Zeichenkette1	Alphanumerische Zeichenfolge, welche verglichen werden soll. Format: Script-Literal oder Script-Variable
Zeichenkette2	Alphanumerische Zeichenfolge, welche verglichen werden soll. Format: Script-Literal oder Script-Variable
Wildcard1	Wildcardzeichen für beliebige Zeichen. Format: Script-Literal oder Script-Variable Standard: "*"
Wildcard2	Wildcardzeichen für genau ein Zeichen. Format: Script-Literal oder Script-Variable Standard: "_"

Rückgabewerte
"Y" - Die beiden Zeichenketten sind identisch. "N" - Die beiden Zeichenketten sind unterschiedlich.

Anmerkungen

Die Script-Funktion vergleicht, ob *Zeichenkette2* mit *Zeichenkette1* identisch ist. Dabei wird zwischen Groß- und Kleinschreibung unterschieden (case-sensitive).

In *Zeichenkette2* können Wildcardzeichen verwendet werden, um ein Raster für den Vergleich zu bilden. Standardmäßig steht "*" dabei für beliebige, "_" für genau ein Zeichen.

Es können auch andere Wildcardzeichen verwendet und *Wildcard1* und/oder *Wildcard2* zugewiesen werden.

Beispiele

Das erste Beispiel gibt das Ergebnis "N" im Aktivierungsprotokoll aus.

```
:SET &RET# = STR_MATCH("UserInterface", "User-Interface")
 :PRINT &RET#
```

Das zweite Beispiel verwendet Wildcardzeichen zum Vergleich. Das Ergebnis "Y" wird im Aktivierungsprotokoll ausgegeben.

```
:SET &RET# = STR_MATCH("UserInterface", "U*I*")
 :PRINT &RET#
```

Das dritte Beispiel verwendet ein Wildcardzeichen, welches explizit festgelegt wird. Das Ergebnis "Y" wird im Aktivierungsprotokoll ausgegeben.

```
:SET &RET# = STR_MATCH("UserInterface", "User#", "#")
 :PRINT &RET#
```

Im vierten Beispiel wird ebenfalls ein Wildcardzeichen verwendet. Dieses steht jedoch für genau ein Zeichen. Deshalb lautet das Ergebnis "N".

```
:SET &RET# = STR_MATCH("UserInterface", "User#",, "#")
 :PRINT &RET#
```

Siehe auch:

Script-Sprachmittel - Strings
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.19 STR_PAD

Script-Funktion: Zeichenkette auf eine bestimmte Länge erweitern.

Syntax

STR_PAD*(Zeichenkette, *[Zeichen]*, ***Gesamtlänge, Ausrichtung)***

Syntaxteil	Beschreibung/Format
Zeichenkette	Zeichenkette, die erweitert werden soll. Format: Script-Literal oder Script-Variable
Zeichen	Beliebiges Zeichen, das für die Erweiterung verwendet wird. Format: AE-Name, Script-Literal oder Script-Variable Standardwert: " "
Gesamtlänge	Länge, auf welche die Zeichenkette erweitert werden soll. Format: Zahl ohne Hochkommata, Script-Literal oder Script-Variable
Ausrichtung	Ausrichtung der Zeichenkette Format: Script-Literal oder Script-Variable Erlaubte Werte: "LEFT" - Zeichenkette wird links angeordnet "RIGHT" - Zeichenkette befindet sich rechts "CENTER" - Zeichenkette mittig positionieren

Rückgabewert
Erweiterte Zeichenkette

Anmerkungen

Die Script-Funktion vergrößert eine Zeichenkette auf eine bestimmte Länge. Für die Erweiterung wird dabei jenes Zeichen verwendet, das beim Parameter *Zeichen* angegeben ist. Standardmäßig wird der neue Platz mit Leerzeichen aufgefüllt.

Weiters kann definiert werden, wie die Zeichenkette im Ergebnis angeordnet sein soll (Parameter *Ausrichtung*). Möglich ist dabei rechts, links oder mittig.
Bei mittiger Anordnung werden links und rechts von der Zeichenkette soviele Zeichen platziert, dass die gewünschte Länge erreicht wird. Bei einer ungeraden Anzahl von aufzufüllenden Zeichen, wird rechts vom String ein Zeichen mehr eingefügt.

Beispiel

Folgendes Beispiel vergrößert die selbe Zeichenkette auf 3 unterschiedliche Arten und gibt das Ergebnis im Aktivierungsprotokoll aus.

```
:SET &STRING# = "Test String"
:SET &STRLEN# = 15
:SET &STRNEW# = STR_PAD(&STRING#, ".", &STRLEN#, "LEFT")
:PRINT &STRNEW#
:SET &STRNEW# = STR_PAD(&STRING#,, &STRLEN#, "CENTER")
:PRINT &STRNEW#
:SET &STRNEW# = STR_PAD(&STRING#, "_", &STRLEN#, "RIGHT")
:PRINT &STRNEW#
```

Ergebnis im Aktivierungsprotokoll:

```
Test String....
  Test String
____Test String
```

Siehe auch:

Script-Sprachmittel	Beschreibung
STR_SPLIT	Zeichenkette mit Hilfe eines Trennzeichens in mehrere Teile aufspalten.

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.20 STR_REVERSE

Script-Funktion: Kehrt die Reihenfolge der Zeichen einer Zeichenkette um.

Syntax

STR_REVERSE*(Zeichenkette)*

Syntaxteil	Beschreibung/Format
Zeichenkette	Alphanumerische Zeichenfolge, welche bearbeitet werden soll. Format: Script-Literal oder Script-Variable

Rückgabewert
Zeichenkette deren Zeichen in umgekehrter Reihenfolge sind

Beispiel

Das Beispiel gibt das Ergebnis "EA" im Aktivierungsprotokoll aus.

```
:SET &RET# = STR_REVERSE("AE")
:PRINT &RET#
```

Siehe auch:

Script-Sprachmittel - Strings
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.21 STR_RTRIM

Script-Funktion: Entfernt Leerzeichen am Ende einer Zeichenkette.

Syntax

STR_RTRIM*(Zeichenkette)*

Syntaxteil	Beschreibung/Format
Zeichenkette	Alphanumerische Zeichenfolge, welche bearbeitet werden soll. Format: Script-Literal oder Script-Variable

Rückgabewert
Zeichenkette ohne abschließende Leerzeichen

Beispiel

Das Beispiel gibt das Ergebnis (" Datenbank wird geöffnet...") im Aktivierungsprotokoll aus.

```
:SET &RET# = STR_RTRIM("   Datenbank wird geöffnet...   ")
 :PRINT &RET#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
STR_LTRIM	Entfernt Leerzeichen am Anfang einer Zeichenkette.
STR_TRIM	Entfernt Leerzeichen am Anfang und Ende einer Zeichenkette.

Script-Sprachmittel - Strings
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.22 STR_SPLIT

Script-Funktion: Zeichenkette mit Hilfe eines Trennzeichens in mehrere Teile aufspalten.

Syntax

STR_SPLIT*(Zeichenkette, Trennzeichen)*

Syntaxteil	Beschreibung/Format
Zeichenkette	Zeichenkette, die aufgeteilt werden soll. Format: Script-Literal oder Script-Variable
Trennzeichen	Beliebiges Zeichen Format: AE-Name, Script-Literal oder Script-Variable
Rückgabewert	
Array, welches die einzelnen Teil-Strings enthält.	

Anmerkungen

Die Script-Funktion teilt die angegebene *Zeichenkette* mit Hilfe eines bestimmten *Trennzeichens* in mehrere Teil-Strings auf. Als Ergebnis wird ein Array zurückgeliefert, in dem die einzelnen Zeichenketten gespeichert sind.

Das Trennzeichen selbst ist im Ergebnis nicht enthalten.

Um das Ergebnis zu speichern, definieren Sie zuerst ein Script-Array mittels :DEFINE und verwenden Sie anschließend das Sprachmittel :FILL. Achten Sie bei der Definition des Arrays, dass der Datentyp "string" verwendet wird.

Beispiele

Folgendes Beispiel teilt die Zeichenkette "123_456_789" gemäß dem Trennzeichen "_" auf und speichert das Ergebnis in ein String-Array. Anschließend werden die einzelnen Array-Elemente im Aktivierungsreport ausgegeben.

```
:DEFINE &STRINGS#, string, 5
:SET &STRING# = "123_456_789"
:FILL&STRINGS#[] = STR_SPLIT(&STRING#,"_")
```

```
:SET&VAR# = 1
:SET&LEN# = LENGTH(&STRINGS#[])

:WHILE &VAR# LE &LEN#
:SET &VAR# = FORMAT(&VAR#,"0")
:PRINT "&VAR#. Teilstring = &STRINGS#[&VAR#]"
:SET&VAR# = &VAR#+ 1
:ENDWHILE
```

Ausgabe im Aktivierungsprotokoll:

```
2013-10-23 10:30:48 - U0020408 1. Teilstring = 123
2013-10-23 10:30:48 - U0020408 2. Teilstring = 456
2013-10-23 10:30:48 - U0020408 3. Teilstring = 789
2013-10-23 10:30:48 - U0020408 4. Teilstring =
2013-10-23 10:30:48 - U0020408 5. Teilstring =
```

Siehe auch:

Script-Sprachmittel	Beschreibung
:DEFINE	Deklaration einer Script-Variable mit einem bestimmten Datentyp.
:FILL	Speichert mehrere Werte in ein Script-Array.
:PUBLISH	Script-Variablen und Arrays als Objektvariablen definieren.
STR_PAD	Zeichenkette auf eine bestimmte Länge erweitern.

Script-Sprachmittel - Aktivierungsdaten
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.23 STR_STARTS_WITH

Script-Funktion: Prüft ob eine Zeichenkette mit einer bestimmten anderen Zeichenkette beginnt.

Syntax

STR_STARTS_WITH*(Zeichenkette1, Zeichenkette2)*

Syntaxteil	Beschreibung/Format
Zeichenkette1	Zeichenfolge, die geprüft werden soll. Format: Script-Literal oder Script-Variable
Zeichenkette2	Zeichenfolge, welche zur Prüfung verwendet wird. Format: Script-Literal oder Script-Variable
Rückgabewert	
"Y" = Zeichenkette1 beginnt mit Zeichenkette2 "N" = Zeichenkette beginnt nicht mit Zeichenkette2	

Anmerkungen

Das Sprachmittel prüft ob der Beginn einer bestimmten Zeichenkette einer anderen Zeichenkette gleicht. Anzugeben ist dabei die zu prüfende Zeichenkette (Parameter *Zeichenkette1*) und der Beginn, ebenfalls in Form einer Zeichenkette (Parameter *Zeichenkette2*).

Die Script-Funktion liefert auch "Y", wenn *Zeichenkette1* und *Zeichenkette2* ident sind.

Beachten Sie, dass Klein- und Großschreibung beim Vergleich beachtet wird.

Beispiel

Folgendes Beispile liest den Objekttitel aus und prüft ob dieser mit dem Begriff "Docu" beginnt. Falls ja, wird eine entsprechende Meldung im Aktivierungsreport protokolliert.

```
:SET &VAR# = GET_ATT(OBJECT_TITLE)
:SET &CHECK# = STR_STARTS_WITH(&VAR#,"Docu")
:IF &CHECK# EQ "Y"
: P "Objekttitel beginnt mit 'Docu'"
:ENDIF
```

Siehe auch:

Script-Sprachmittel	Beschreibung
STR_ENDS_WITH	Prüft ob eine Zeichenkette mit einer bestimmten anderen Zeichenkette endet.

Script-Sprachmittel - Strings
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.24 STR_SUBSTITUTE

Script-Funktion: Ersetzt Zeichen oder Zeichenkette innerhalb einer Zeichenkette.

Syntax

STR_SUB[STITUTE] ***(Zeichenkette,*** *[Alt] [,Neu]****)***

Syntaxteil	Beschreibung/Format
Zeichenkette	Alphanumerische Zeichenfolge, innerhalb der ersetzt werden soll. Format: Script-Literal oder Script-Variable
Alt	Alphanumerische Zeichenfolge, die in der Zeichenkette ersetzt werden soll. Format: Script-Literal oder Script-Variable Standardwert: " "

Neu	Alphanumerische Zeichenfolge, die *Alt* ersetzen soll. Format: Script-Literal oder Script-Variable Standardwert: " "

Rückgabewert
Zeichenkette, die durch die Ersetzung entstanden ist.

Anmerkungen

Die Script-Funktion ersetzt ein Zeichen oder eine Zeichenkette innerhalb einer Zeichenkette.

Die Parameter *Alt* und *Neu* sind optional und besitzen kein Größenlimit. Wurde *Alt* nicht als Parameter angegeben, wird jedes einzelne Leerzeichen durch das Zeichen bzw. die Zeichenkette von *Neu* ersetzt. Wurde *Neu* nicht als Parameter verwendet, wird *Alt* durch ein einziges Leerzeichen ersetzt.

Kommt *Alt* in der *Zeichenkette* nicht vor, gibt die Funktion die Zeichenkette unverändert zurück.

Die Script-Funktion kann auch dazu verwendet werden, *Alt* aus der Zeichenkette zu löschen. Dafür müssen als *Neu* zwei aufeinanderfolgende Hochkommata (ohne Leerzeichen) angegeben werden.

Beispiel

Im ersten Beispiel wird das Zeichen "A" durch das Zeichen "B" ersetzt. Das Ergebnis "BBBBB" wird im Aktivierungsprotokoll ausgegeben.

```
:SET &RET# = STR_SUBSTITUTE ("AAAAA", "A", "B")
:PRINT  &RET#
```

Das zweite Beispiel ersetzt die Zeichenkette "AAAAA" durch das Zeichen "B". Das Ergebnis "B" wird im Aktivierungsprotokoll ausgegeben.

```
:SET &RET# = STR_SUBSTITUTE ("AAAAA", "AAAAA", "B")
:PRINT  &RET#
```

Im dritten Beispiel wird die Zeichenkette "AA" durch die Zeichenkette "BB" ersetzt. Das Ergebnis "BBBBA" wird im Aktivierungsprotokoll ausgegeben.

```
:SET &STR1# = "AA"
:SET  &STR2# = "BB"
:SET  &RET# = STR_SUB ("AAAAA", &STR1#, &STR2#)
:PRINT  &RET#
```

Das vierte Beispiel löscht die Leerzeichen aus der Zeichenkette. Das Ergebnis "AE" wird im Aktivierungsprotokoll ausgegeben.

```
:SET &RET# = STR_SUB ("A E", " ", "")
:PRINT  &RET#
```

Siehe auch:

Script-Sprachmittel - Strings
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.25 STR_SUBSTITUTE_VAR, STR_SUB_VAR

Script-Funktion: Ersetzt den Namen von Script-Variablen durch deren Wert.

Syntax

STR_SUB[STITUTE]_VAR*(Variablen)*

Syntaxteil	Beschreibung/Format
Variablen	Script-Variable oder Zeichenkette, die einen oder mehrere Variablennamen enthält Format: Script-Literal oder Script-Variable

Rückgabewert
Zeichenkette mit den Werten der Script-Variablen

Anmerkungen

Das Sprachmittel STR_SUB_VAR bietet die Möglichkeit einen oder mehrere Namen von Script-Variablen, welche wiederum als Zeichenkette in einer anderen Script-Variable gespeichert sind, durch den tatsächlichen Variablen-Wert zu ersetzen. Die folgenden Beispiele sollen das Verhalten verdeutlichen.

Beim Sprachmittel GET_PROCESS_LINE gibt es einen Parameter, der ebenfalls die Bezeichnung STR_SUB[STITUTE]_VAR trägt und die selbe Ersetzung für die Zeile einer Datensequenz durchführt.

Beispiel

In folgendem Beispiel wird der Name einer Script-Variable (&VAR#) in eine andere Variable gespeichert (&VAR_NOSUB#), indem das Zeichen & doppelt angegeben wird.

Wird &VAR_NOSUB# ausgegeben, so wird der Name der Script-Variable &VAR# in das Aktivierungsprotokoll geschrieben. Um diesen ebenfalls durch den Variablenwert zu ersetzen, ist das Sprachmittel STR_SUB_VAR zu verwenden.

```
:SET &VAR# = "Script-Variable"
:SET &VAR_NOSUB# = "&&VAR# = &VAR#"
:PRINT &VAR_NOSUB#
:SET &VAR_SUB# = STR_SUB_VAR(&VAR_NOSUB#)
:PRINT &VAR_SUB#
```

Ausgabe im Aktivierungsprotokoll:

```
2011-05-06 10:34:04 - U0020408 &VAR# = Script-Variable
2011-05-06 10:34:04 - U0020408 Script-Variable = Script-Variable
```

Im zweiten Beispiel wird ein Wert aus dem Variablen-Objekt VARA.SUB ermittelt, der wiederum den Namen von 2 Script-Variablen enthält. Diese 2 Script-Variablen werden anschließend erstellt und gesetzt.

Wird der Wert des Variablen-Objektes direkt ausgegeben, so werden nur die Namen der Script-Variablen in den Report geschrieben. Es erfolgt keine direkte Wertersetzung.

Um die Namen der 2 Script-Variablen durch deren Wert zu ersetzen, wird das Sprachmittel STR_SUB_VAR benötigt.

```
:SET &VARA# = GET_VAR(VARA.SUB, "SUBVAR")
:SET &VAR1# = "Hello"
:SET &VAR2# = "World"
:PRINT "Inhalt ohne Ersetzungen: &VARA#"
:SET &VARA_SUB_VAR# = STR_SUB_VAR(&VARA#)
:PRINT "Inhalt mit Ersetzungen: &VARA_SUB_VAR#"
```

Ausgabe zuerst ohne und danach mit STR_SUB_VAR:

```
2011-05-06 10:34:04 - U0020408 Inhalt ohne Ersetzungen: &VAR1# &VAR2#
2011-05-06 10:34:04 - U0020408 Inhalt mit Ersetzungen: Hello World
```

Siehe auch:

Sprachmittel	Beschreibung
GET_PROCESS_LINE	Ermittelt den aktuellen Zeileninhalt einer Datensequenz.

Script-Sprachmittel - Objekte aktivieren
Beispielsammlung
Benachrichtigung mit variablem Meldungstext
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.26 STR_TRIM

Script-Funktion: Entfernt Leerzeichen am Anfang und Ende einer Zeichenkette.

Syntax

STR_TRIM*(Zeichenkette)*

Syntaxteil	Beschreibung/Format
Zeichenkette	Alphanumerische Zeichenfolge, welche bearbeitet werden soll. Format: Script-Literal oder Script-Variable

Rückgabewert
Zeichenkette ohne führende und abschließende Leerzeichen

Beispiel

Das Beispiel gibt das Ergebnis ("Datenbank wird geöffnet...") im Aktivierungsprotokoll aus.

```
:SET &RET# = STR_TRIM("   Datenbank wird geöffnet...   ")
 :PRINT &RET#
```

Siehe auch:

Script-Sprachmittel	Beschreibung
STR_LTRIM	Entfernt Leerzeichen am Anfang einer Zeichenkette.
STR_RTRIM	Entfernt Leerzeichen am Ende einer Zeichenkette.

Script-Sprachmittel - Strings
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.27 STR_UC, CONV_UC

Script-Funktionen: Setzen alle Kleinbuchstaben einer Zeichenkette auf Großbuchstaben um.

Syntax

STR_UC*(Zeichenkette)*
CONV_UC*(Zeichenkette)*

Syntaxteil	Beschreibung/Format
Zeichenkette	Alphanumerische Zeichenfolge Format: Script-Literal oder Script-Variable

Rückgabewert
Zeichenkette in der alle Buchstaben Großbuchstaben sind.

Anmerkungen

Beide Script-Funktionen sind funktional identisch.

Beispiele

Beide Beispiele setzen eine *Zeichenkette* in Großbuchstaben um. Die *Zeichenkette* wird der Funktion im ersten Beispiel als String-Literal, im zweiten als Script-Variable übergeben. Die Ergebnisse der Funktion lauten "ABCDEFGH 123&%$§" sowie "BITTE SICHERUNG STARTEN!".

```
:SET &STRING# = CONV_UC('abcdefgh 123&%$§')
```

```
:SET &MSG# = "Bitte Sicherung starten!"
 :SET &STRING# = STR_UC(&MSG#)
```

Siehe auch:

Script-Sprachmittel	Beschreibung
CONV_LC oder STR_LC	Setzen alle Großbuchstaben einer Zeichenkette auf Kleinbuchstaben um.

Script-Sprachmittel - Strings
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

3.14.28 UC_CRLF

Script-Funktion: Liefert einen Zeilenumbruch.

Syntax

UC_CRLF*()*

Beispiele

Im ersten Beispiel wird die Meldung der Benachrichtigung zweizeilig ausgegeben.

```
:PUT_ATT CALL_TEXT = "Der Workflow MAWI.TAG ist blockiert."
:PUT_ATT_APPEND CALL_TEXT = UC_CRLF()
:PUT_ATT_APPEND CALL_TEXT = "Eskalation in 10 Minuten."
```

Im zweiten Beispiel wird die gleiche Meldung ausgeben. Der Zeilenumbruch wird aber über eine Script-Variable realisiert.

```
:SET  &NL#=UC_CRLF()
:PUT_ATT CALL_TEXT = "Der Workflow MAWI.TAG ist
blockiert.&NL#Eskalation in 10 Minuten."
```

Natürlich kann die Script-Funktion auch für Texte, die Sie per E-Mail senden, verwendet werden wie das dritte Beispiel zeigt.

```
:SET &NL# = UC_CRLF()
:SET &TEXT# = "Der Workflow MAWI.TAG ist blockiert.&NL#Eskalation in
10 Minuten."
:SET &RET# = SEND_MAIL("meier@automic.at",,"Aufgabe konnte nicht
gestartet werden!",&TEXT#)
```

Siehe auch:

Script-Sprachmittel - Fehlerbehandlung und Meldungen
Beispielsammlung
Ermittlung der Fehlermeldung und -nummer
Allgemeines zum Script
Script-Sprachmittel - alphabetische Auflistung
Script-Sprachmittel - funktionelle Gliederung

4 AE JCL für Anwendungen

4.1 Allgemeines zur AE JCL für Anwendungen

AE stellt nicht nur Job-Objekte für Betriebssysteme, sondern auch für Standard Software Lösungen wie z.B. SAP, PeopleSoft, Oracle Applications und Siebel zur Verfügung. Da letztere keine JCL (Job Control Language) besitzen sondern Schnittstellen bieten, gibt es im AE Script eine Reihe von speziellen Funktionen, um Verarbeitungsanweisungen durchzuführen.

Diese Funktionen werden mit Hilfe einer grafischen Oberfläche (FORMS) erstellt, die auch eine Onlineverbindung zum jeweiligen Standardsystem unterhalten; damit ist die Möglichkeit gegeben beispielsweise mit DropDown Listen zu arbeiten, um Attribute für die Funktionen auszuwählen (Druckername, usw.).

Alternativ dazu können die Anweisungen auch in der klassischen Form in den Script-Registerkarten des Job-Objektes hinterlegt werden (Script-Editor).

Siehe auch:

Allgemeines zum Script
Registerkarte **Form** (SAP)
Registerkarte **Form** (PeopleSoft)

4.2 Oracle Applications

4.2.1 Allgemeines zur Oracle Applications-JCL

Folgende Übersicht zeigt die JCL Script-Sprachmittel für Oracle Applications und die zugehörigen Schnittstellen.

Script-Sprachmittel

Script-Sprachmittel	Beschreibung
OA_ADD_LAYOUT	Fügt einem Request ein Layout hinzu.
OA_ADD_ NOTIFICATION	Fügt einem Request eine Benachrichtigung hinzu.
OA_ADD_PRINTER	Fügt einem Request einen zusätzlichen Drucker hinzu.
OA_SET_PRINT_ DEFAULTS	Setzt Standardwerte für die Druckparameter, die beim Ausführen von Concurrent Programs verwendet werden.
OA_SUBMIT_ REQUEST	Schickt eine Anforderung zum Ausführen eines Concurrent Programs in Oracle Applications.

Siehe auch:

Allgemeines zur AE JCL für Anwendungen

4.2.2 OA_ADD_LAYOUT

Fügt einem Request ein Layout hinzu.

Syntax

OA_ADD_LAYOUT

[**APPLICATION**=...]
[**,NAME**=...]
[**,LANGUAGE**=...]
[**,TERRITORY**=...]
[**,OUTPUTFORMAT**=...]

Syntaxteil	Beschreibung/Format
APPLICATION=	Kurzname der Vorlagen-Applikation Format: Script-Literal
NAME=	Bezeichnung des Templates Format: Script-Literal
LANGUAGE=	Sprache der Templatedatei Format: Script-Literal
TERRITORY=	Gebietsangabe der Templatedatei Format: Script-Literal
OUTPUTFORMAT=	Ausgabeformat Format: Script-Literal

Anmerkungen

Rufen Sie im Script für jedes Layout die Anweisung OA_ADD_LAYOUT auf. Sobald ein OA_SUBMIT_REQUEST durchgeführt wird, übergibt der Agent die Angaben an Oracle. Kommen weitere Submits im Script vor, so müssen Sie die Layouts zuvor erneut mit OA_ADD_LAYOUT-Anweisungen definieren.

Tritt bei der Übergabe eines Layouts ein Fehler auf, so wird dieser im Aktivierungsreport protokolliert und der Job mit dem Rückgabewert 10 beendet.

Beachten Sie, dass der Einsatz von OA_ADD_LAYOUT mindestens die Oracle Applications Version 11.5.10.2 erfordert!

Beispiel

```
OA_ADD_LAYOUT APPLICATION='FND',NAME='XXPOST_XXPPOPS_
PP2',LANGUAGE='en',TERRITORY='US',OUTPUTFORMAT='PDF'
```

Siehe auch:

Allgemeines zur Oracle Applications-JCL

4.2.3 OA_ADD_NOTIFICATION

Fügt einem Request eine Benachrichtigung hinzu.

Syntax

OA_ADD_NOTIFICATION

USER=...

Syntaxteil	Beschreibung/Format
USER=	Name des Benutzers Format: Script-Literal

Anmerkungen

Rufen Sie im Script für jede Benachrichtigung die Anweisung OA_ADD_NOTIFICATION auf. Sobald ein OA_SUBMIT_REQUEST durchgeführt wird, übergibt der Agent die Angaben an Oracle. Kommen weitere Submits im Script vor, so müssen Sie die Benachrichtigungen zuvor erneut mit OA_ADD_NOTIFICATION-Anweisungen definieren.

Tritt bei der Übergabe einer Benachrichtigung ein Fehler auf, so wird dieser im Aktivierungsreport protokolliert und der Job mit dem Rückgabewert 10 beendet.

Beispiel

```
OA_ADD_NOTIFICATION USER='MAX MEIER'
```

Siehe auch:

Allgemeines zur Oracle Applications-JCL

4.2.4 OA_ADD_PRINTER

Fügt einem Request einen zusätzlichen Drucker hinzu.

Syntax

OA_ADD_PRINTER

PRINTER=...
[**,COPIES**=...]

Syntaxteil	Beschreibung/Format
PRINTER=	Name des Druckers Format: Script-Literal
COPIES=	Anzahl der Kopien Format: Zahl Standardwert: "0"

Anmerkungen

Mit der Anweisung OA_SET_PRINT_DEFAULTS können Sie nur einen Drucker definieren. Verwenden Sie daher OA_ADD_PRINTER, wenn Sie mehrere angeben möchten.

Rufen Sie zu diesem Zweck im Script für jeden Drucker die Anweisung auf. Sobald ein OA_SUBMIT_REQUEST durchgeführt wird, übergibt der Agent die Druckerangaben an Oracle. Kommen weitere Submits im Script vor, so müssen Sie die Drucker zuvor erneut mit OA_ADD_PRINTER-Anweisungen definieren.

Der Drucker, den Sie mit OA_SET_PRINT_DEFAULTS bestimmen, gilt hingegen für alle folgenden OA_SUBMIT_REQUEST-Aufrufe bis zum Job-Ende oder bis zur nächsten OA_SET_PRINT_DEFAULTS-Anweisung.

Tritt bei der Übergabe eines Druckers ein Fehler auf, so wird dieser im Aktivierungsreport protokolliert und der Job mit dem Rückgabewert 10 beendet.

Beispiel

```
OA_ADD_PRINTER PRINTER='HPLJ',COPIES=2
```

Siehe auch:

Allgemeines zur Oracle Applications-JCL

4.2.5 OA_SET_PRINT_DEFAULTS

Setzt Standardwerte für die Druckparameter, die beim Ausführen von Concurrent Programs verwendet werden.

Syntax

OA_SET_PRINT_DEFAULTS

PRINTER=...
[**,STYLE**=...]
[**,COPIES**=...]

Syntaxteil	Beschreibung/Format
PRINTER=	Name für Drucker Format: Script-Literal
STYLE=	Stil Format: Script-Literal
COPIES=	Anzahl der Kopien Format: Zahl Standardwert: "0"

Anmerkungen

Mit dem Script-Sprachmittel können Druckparametern Standardwerte zugewiesen werden. Sie gelten für das gesamte Script oder solange, bis im Script ein neuer Standardwert zugewiesen wird. Die Defaults sind nur für den Job gültig, in dem sie gesetzt wurden.

Durch das Setzen von Standardwerten wird die Übersichtlichkeit des AE Scripts erhöht. Gleichzeitig werden Probleme mit der Beschränkung von 255 Zeichen pro Script-Zeile vermieden, da die eigentliche Verarbeitungsanweisung erst im Agenten für Oracle Applications zusammengesetzt wird.

Das Script-Sprachmittel wird für OA_SUBMIT_REQUEST genutzt, um den Druckparametern Standardwerte zuzuweisen.

Beispiel

Im Beispiel werden Standardwerte für das Ausgabegerät und die Anzahl der Ausdrucke gesetzt.

```
OA_SET_PRINT_DEFAULTS PRINTER='MINE'
OA_SET_PRINT_DEFAULTS COPIES=2
OA_SUBMIT_REQUEST
APPLICATION='FND',PROGRAM='MYPROG',DESCRIPTION='Sonntagslauf von
UC4',ARG1='602'
```

Siehe auch:

Allgemeines zur Oracle Applications-JCL

4.2.6 OA_SUBMIT_REQUEST

Schickt eine Anforderung zum Ausführen eines Concurrent Programs in Oracle Applications.

Syntax

OA_SUBMIT_REQUEST

APPLICATION=...
,PROGRAM=...
[**,DESCRIPTION**=...]
[**,ARG1**=...[,**ARGn**=...]]

Syntaxteil	Beschreibung/Format
APPLICATION=	Name der Anwendung, zu welchem das auszuführende Programm gehört Format: Script-Literal
PROGRAM=	Name des Concurrent Programs, welches ausgeführt werden soll Format: Script-Literal
DESCRIPTION=	Beschreibung zum Concurrent Process. Format: Script-Literal Diese Beschreibung wird online im Concurrent Request Form angezeigt.
ARG1= **:** **ARGn=**	Beliebige Parameter, die an das Concurrent Program übergeben werden. Das können Funktionen, Konstanten, Schlüsselworte etc.sein. Beispiele: chr(0), NULL, 'abc', NAME_IN('ORDERS.ORDER_ID'), 123 Format: Script-Literal

Anmerkungen

Das Script-Sprachmittel erzeugt eine Prozessanforderung für ein Concurrent Program in Oracle Applications und überwacht die Durchführung. Das Ende des Concurrent Process erkennt AE anhand der Phase und des zugehörigen Status.

⚠ Die Parameter für ARG1 bis ARG*n* müssen unter doppelten Anführungszeichen gefolgt von einfachen Anführungszeichen stehen, wenn sie keine Variablen sind.

Beispiel

Im Beispiel soll das Concurrent Program "MYPROG" in Oracle Applications ausgeführt werden. Zusätzlich werden die Beschreibung und ein weiterer Parameter übergeben.

```
OA_SUBMIT_REQUEST
APPLICATION='FND',PROGRAM='MYPROG',DESCRIPTION="'Sonntagslauf von
UC4'",ARG1="'602'"
```

Das folgende Beispiel zeigt, dass für die Anwendung von Anführungszeichen beim Parameter ARG1.

```
OA_SUBMIT_REQUEST
APPLICATION='FND',PROGRAM='FNDPRNEV',ARG1="'PRINTER'"
```

Siehe auch:

Allgemeines zur Oracle Applications-JCL

4.3 PeopleSoft

4.3.1 Allgemeines zur PeopleSoft-JCL

Folgende Übersicht zeigt die JCL Script-Sprachmittel für PeopleSoft und die zugehörigen Schnittstellen.

Script-Sprachmittel und verwendete Schnittstellen

Script-Sprachmittel	Beschreibung	Schnittstelle
PS_GET_HEARTBEAT	Überwacht einen PeopleSoft Process Scheduler Server.	PROCESSREQUEST_SBB
PS_GRANT_OUTPUT_ACCESS	Berechtigt Benutzer oder Rollen für den Zugriff auf den Output eines PeopleSoft-Prozesses.	PROCESSREQUEST_SBB
PS_MODIFY_RUNCONTROL	Ändert einzelne Parameter in Run Controls.	PROCESSREQUEST_SBB
PS_RUN_JOB	Startet und überwacht einen PeopleSoft-Job.	PROCESSREQUEST_SBB
PS_RUN_PROCESS	Startet und überwacht einen PeopleSoft-Prozess.	keine, PROCESSREQUEST oder PROCESSREQUEST_SBB (abhängig von den verwendeten Parametern)
PS_SET_BINDVAR	Ersetzt den Wert einer Bind-Variablen in einer Prozessdefinition.	PROCESSREQUEST_SBB

Siehe auch:

Allgemeines zur AE JCL für Anwendungen

4.3.2 PS_GET_HEARTBEAT

Überwacht einen PeopleSoft Process Scheduler Server.

Schnittstelle: PROCESSREQUEST_SBB
PeopleSoft-Version: 8+

Syntax

PS_GET_HEARTBEAT

RUNLOCATION=...
[**,MAXPERIOD**=...]

Syntaxteil	Beschreibung/Format
RUNLOCATION=	Name eines zugewiesenen PeopleSoft Process Scheduler Batch Servers, beispielsweise PSUNX oder PSNT Format: Script-Literal
MAXPERIOD=	Maximale Zeitspanne in Sekunden, welche zwischen der Systemzeit des PeopleSoft Datenbank-Servers und der Zeit des letzten Heartbeats vom Process Scheduler Server erlaubt ist. Format: Zahl

Anmerkungen

Jeder Process Scheduler Server meldet sich zyklisch an der PeopleSoft-Datenbank. Dieser Vorgang wird als Heartbeat des Process Scheduler Servers bezeichnet.

Das Script-Sprachmittel PS_GET_HEARTBEAT ermöglicht es, die Verfügbarkeit des Process Scheduler Servers zu überwachen. Es ermittelt die Systemzeit des PeopleSoft Datenbank-Servers und die Zeit des letzten Heartbeats vom angegebenen Process Scheduler Server. Beide Zeiten werden dann im Aktivierungsreport des AE Jobs ausgegeben. Die Meldung mit der Meldungsnummer U2004942 enthält erstens die Zeit des Datenbank-Servers und zweitens die Zeit des letzten Heartbeats. Diese Information kann im Post-Script ausgewertet werden.

Wird der optionale Parameter MAXPERIOD verwendet, wird die Zeitdifferenz zwischen PeopleSoft Datenbank-Server und letztem Heartbeat errechnet und mit der hier angegebenen Zeitspanne verglichen. Ist die errechnete Zeitdifferenz größer als die maximal erlaubte Zeitspanne, bricht der AE Job ab. Wird die maximal erlaubte Zeitspanne durch die errechnete Zeitdifferenzen nicht erreicht oder überschritten, endet der AE Job normal.

Beispiele

Im ersten Beispiel wird die Systemzeit des PeopleSoft Datenbank-Servers und die Zeit des letzten Heartbeats vom Process Scheduler Server "PSNT" ermittelt und im Aktivierungsreport ausgegeben.

```
PS_GET_HEARTBEAT RUNLOCATION='PSNT'
```

Das zweite Beispiel vergleicht die Zeitdifferenz zwischen PeopleSoft Datenbank-Server und letztem Heartbeat von "PSNT" mit der maximal erlaubten Zeitspanne. Der letzte Heartbeat darf maximal 20 Sekunden älter als die Zeit des PeopleSoft Datenbank-Servers sein, ansonsten bricht der Job ab.

```
PS_GET_HEARTBEAT RUNLOCATION='PSNT', MAXPERIOD=20
```

Siehe auch:

Allgemeines zur PeopleSoft-JCL

4.3.3 PS_GRANT_OUTPUT_ACCESS

Berechtigt Benutzer oder Rollen für den Zugriff auf den Output eines PeopleSoft-Prozesses.

Schnittstelle: PROCESSREQUEST_SBB
PeopleSoft-Version: 8+

Syntax

PS_GRANT_OUTPUT_ACCESS

NAME=...
[**,TYPE=**...]
[**,PID=**...]
[**,JOBITEM=**...]

Syntaxteil	Beschreibung/Format
NAME=	Berechtigungsidentifikation: Name einer Rolle oder eines Benutzers Format: Script-Literal
TYPE=	Berechtigungstyp: Rolle oder Benutzer Format: Script-Literal Erlaubte Werte: "R" (Standard) und "U" "R" - Berechtigung wird für eine Rolle erteilt. "U" - Berechtigung wird für einen Benutzer erteilt. Geben Sie etwas anderes als "R" oder "U" an, verwendet der Agent automatisch "R".
PID=	Prozessinstanz-Nummer Format: Zahl
JOBITEM=	Item-Nummer eines Prozesses innerhalb eines PeopleSoft-Jobs Format: Zahl

Anmerkungen

Das Script-Sprachmittel wird verwendet, um nach der Durchführung eines PeopleSoft-Prozesses Berechtigungen für den Zugriff auf dessen Output dynamisch und automatisiert zu vergeben. Berechtigungen können für Rollen oder Benutzer erteilt werden.

Mit dem optionalen Parameter PID= kann ein bestimmter Prozess spezifiziert werden. Wird der Parameter nicht verwendet, bezieht sich das Script-Sprachmittel auf die unmittelbar zuvor gestartete bzw. überwachte Instanz eines Prozesses (PS_RUN_PROCESS) oder eines Jobs (PS_RUN_JOB).

Bezieht sich das Script-Sprachmittel auf einen PeopleSoft-Job, können mit JOBITEM= individuelle Berechtigungen für alle Prozesse dieses Jobs erteilt werden. Mit JOBITEM=1 werden beispielsweise Berechtigungen für den ersten Einzelprozess, mit JOBITEM=2 für den zweiten Einzelprozess usw. vergeben. Falls dieser Parameter nicht angegeben wurde, werden für alle Einzelprozesse des PeopleSoft-Jobs identische Berechtigungen zugewiesen.

Beispiel

Im Beispiel wird dem Benutzer "PS" die Berechtigung für den Zugriff auf den Output eines PeopleSoft-Prozesses erteilt.

```
PS_GRANT_OUTPUT_ACCESS NAME='PS',TYPE='U'
```

Siehe auch:

Allgemeines zur PeopleSoft-JCL

4.3.4 PS_MODIFY_RUNCONTROL

Ändert einzelne Parameter in Run Controls.

Schnittstelle: PROCESSREQUEST_SBB
PeopleSoft-Version: 8+

Syntax

PS_MODIFY_RUNCONTROL

RUNCONTROLID=...
,RECORDNAME=...
,FIELDNAME=...
,FIELDVALUE=...
[**,KEYNAME(1)**=...
,KEYVALUE(1)=...]
[**,KEYNAME(2)**=...
,KEYVALUE(2)=...]
[**,KEYNAME(3)**=...
,KEYVALUE(3)=...]

Syntaxteil	Beschreibung/Format
RUNCONTROLID=	Run Control ID des PeopleSoft-Jobs Format: Script-Literal
RECORDNAME=	Name eines PeopleSoft-Satzes, welcher Teil einer Run Control ist Format: Script-Literal
FIELDNAME=	Name eines Feldes aus dem PeopleSoft-Satz Format: Script-Literal
FIELDVALUE=	Wert, welcher dem Feld zugewiesen werden soll Format: Script-Literal
KEYNAME(*n*)=	Schlüssel in der RunControl-Tabelle Format: Script-Literal n gibt den Level an. Erlaubt sind Werte zwischen 1 und 3
KEYVALUE(*n*)=	Wert für das Schlüsselfeld Format: Script-Literal n gibt den Level an. Erlaubt sind Werte zwischen 1 und 3

Anmerkungen

Mit dem Script-Sprachmittel können Feldern aus PeopleSoft-Sätzen neue Werte zugewiesen und so Parameter von Run Controls direkt geändert werden. Da dabei die PeopleCodes, welche beim Satz und den Feldern definiert sind, nicht durchlaufen werden, ist die Plausibilität der neuen Werte sicherzustellen. So kann das nicht korrekte Zuweisen von Werten auch zu "Inkonsistenzen" führen. Das ist beispielsweise der Fall, wenn ein Von-Datum mit einem neuen Wert versorgt wird, ein dazugehöriges Bis-Datum aber nicht.

Wir empfehlen, um Konflikte zu vermeiden, für die Batchverarbeitung mit AE eigene Run Control ID's anzulegen und zu verwenden.

⚠ Fehlt eine Schlüsselangabe oder ist sie fehlerhaft, so bricht die Modifikation ab und der Job ended abnormal.

Die PeopleSoft-RunControl Tabellen besitzen immer 2 fixe Schlüssel: OPRID und RUN_CNTL_ID. Beachten Sie, dass diese Parameter bei der Verwendung des Sprachmittels nicht anzugeben sind, da sie automatisch generiert werden. Andernfalls werden die Änderungen in der Tabelle nicht durchgeführt.

Beispiel

Im ersten Beispiel wird dem Feld "ASOFDATE" aus dem PeopleSoft-Satz "RUN_CTRL_HR" ein neues Datum zugewiesen.

```
PS_MODIFY_RUNCONTROL RUNCONTROLID='sbb',RECORDNAME='RUN_CNTL_
HR',FIELDNAME='ASOFDATE',FIELDVALUE='20020117'
```

Das zweite Beispiel zeigt die Änderung eines Run Controls für die Währungsumrechnung.

```
PS_MODIFY_RUNCONTROL RUNCONTROLID='myRunControl',RECORDNAME='RUN_CNTL_
CC2_EO',FIELDNAME='RATE_MULT',FIELDVALUE='100',KEYNAME(1)='CURRENCY_
CD',KEYVALUE(1)='AUT'
```

Siehe auch:

Änderungen in Run Controls
Allgemeines zur PeopleSoft-JCL

4.3.5 PS_RUN_JOB

Startet und überwacht einen PeopleSoft-Job.

Schnittstelle: PROCESSREQUEST_SBB
PeopleSoft-Version: 8+

Syntax

PS_RUN_JOB

JOBNAME=...
,RUNCONTROLID=...
[**,RUNLOCATION**=...]

[**,OUTPUTDEST**=...]
[**,OUTDESTTYPE**=...]
[**,OUTDESTFORMAT**=...]

Syntaxteil	Beschreibung/Format
JOBNAME=	Name des PeopleSoft-Jobs, der zu starten und zu überwachen ist Format: Script-Literal
RUNCONTROLID=	Run Control ID des PeopleSoft-Jobs Format: Script-Literal
RUNLOCATION=	Name eines zugewiesenen PeopleSoft Process Scheduler Batch Servers, beispielsweise PSUNX oder PSNT Format: Script-Literal
OUTPUTDEST=	Verzeichnis, in welches der PeopleSoft-Job seine Ausgabe schreibt Format: Script-Literal
OUTDESTTYPE=	Ausgabetyp des PeopleSoft-Jobs, beispielweise Datei, Drucker oder E-Mail Format: Script-Literal Alle gültigen Werte können mit folgender Datenbank-Abfrage angezeigt werden: select XLATSHORTNAME from XLATTABLE where FIELDNAME = 'OUTDESTTYPE';
OUTDESTFORMAT=	Dateiformat, in dem die Ausgabe des PeopleSoft-Jobs gespeichert werden soll, beispielweise TXT, HTM oder PDF Format: Script-Literal Alle gültigen Werte können mit folgender Datenbank-Abfrage angezeigt werden: select XLATSHORTNAME from XLATTABLE where FIELDNAME = 'OUTDESTFORMAT';

Anmerkungen

Das Script-Sprachmittel startet einen PeopleSoft-Job und überwacht dessen Durchführung. Mit einem PeopleSoft-Job können mehrere PeopleSoft-Prozesse nacheinander oder gleichzeitig durchgeführt werden.

Der Name des PeopleSoft-Jobs bezieht sich auf die Definition in PeopleSoft's "Process Scheduler Manager". Ebenso muss der Name des zugewiesenen Process Scheduler Batch Servers einer Definition entsprechen, wenn dieser Parameter verwendet wird.

OUTPUTDEST, OUTDESTTYPE und OUTDESTFORMAT sind weitere optionale Parameter, die verwendet werden können, um die Ausgabe des PeopleSoft-Jobs zu spezifizieren.

Das Script-Sprachmittel kann nur verwendet werden, wenn der Administrator die Schnittstelle PROCESSREQUEST_SBB installiert und in der INI-Datei des Agenten aktiviert hat.

Bind-Variablen können in PS_RUN_JOB nicht verwendet werden.

Beispiel

Im Beispiel werden optionale Parameter dazu verwendet, um die Ausgabe des PeopleSoft-Jobs als SPF-Datei in ein temporäres Verzeichnis zu speichern.

```
PS_RUN_JOB
JOBNAME='3SQR',RUNCONTROLID='sbb',RUNLOCATION=PSNT,OUTDESTTYPE='FILE',
OUTDESTFORMAT='SPF',OUTPUTDEST='c:\temp'
```

Siehe auch:

Allgemeines zur PeopleSoft-JCL

4.3.6 PS_RUN_PROCESS

Startet und überwacht einen PeopleSoft-Prozesses.

Syntax

PS_RUN_PROCESS

PROCESSNAME=...
,PROCESSTYPE=...
,RUNLOCATION=...
,RUNCONTROLID=...
[**,OUTPUTDEST**=...]
[**,OUTDESTTYPE**=...]
[**,OUTDESTFORMAT**=...]

Syntaxteil	Beschreibung/Format
PROCESSNAME=	Name des PeopleSoft-Prozesses, der zu initiieren und zu überwachen ist Format: Script-Literal
PROCESSTYPE=	Prozesstyp des PeopleSoft-Prozesses Format: Script-Literal
RUNLOCATION=	Name eines zugewiesenen PeopleSoft Process Scheduler Batch Servers, beispielsweise PSUNX oder PSNT Format: Script-Literal
RUNCONTROLID=	Run Control ID des PeopleSoft-Prozesses Format: Script-Literal
OUTPUTDEST=	Verzeichnis, in welches der PeopleSoft-Prozess seine Ausgabe schreibt Format: Script-Literal

OUTDESTTYPE=	Ausgabetyp des PeopleSoft-Prozesses, beispielweise Datei, Drucker oder E-Mail Format: Script-Literal Alle gültigen Werte können mit folgender Datenbank-Abfrage angezeigt werden: select XLATSHORTNAME from XLATTABLE where FIELDNAME = 'OUTDESTTYPE'; Schnittstelle: PROCESSREQUEST und PROCESSREQUEST_SBB PeopleSoft-Version: 8+
OUTDESTFORMAT=	Dateiformat, in dem die Ausgabe des PeopleSoft-Prozesses gespeichert werden soll, beispielweise TXT, HTM oder PDF Format: Script-Literal Alle gültigen Werte können mit folgender Datenbank-Abfrage angezeigt werden: select XLATSHORTNAME from XLATTABLE where FIELDNAME = 'OUTDESTFORMAT'; Schnittstelle: PROCESSREQUEST und PROCESSREQUEST_SBB PeopleSoft-Version: 8+

Anmerkungen

Das Script-Sprachmittel erzeugt eine Prozessanforderung für einen PeopleSoft-Prozess und überwacht die Durchführung. Das Ende des PeopleSoft-Prozesses wird anhand des Run Status erkannt, welcher in der PeopleSoft-Datenbank gespeichert wurde.

Name und Typ des PeopleSoft-Prozesses beziehen sich auf die Prozess-Definitionen in PeopleSoft's "Process Scheduler Manager". Ebenso muss der Name des zugewiesenen Process Scheduler Batch Servers einer Definition entsprechen.

OUTPUTDEST, OUTDESTTYPE und OUTDESTFORMAT sind optionale Parameter, die verwendet werden können, um die Ausgabe des PeopleSoft-Prozesses zu spezifizieren.

Einige Parameter des Script-Sprachmittels sind von der eingesetzten PeopleSoft-Version abhängig und in der Syntax-Tabelle gekennzeichnet.

Beispiele

Im ersten Beispiel wird der PeopleSoft-Prozess "THR200" vom Prozesstyp "SQR Report" initiiert.

```
PS_RUN_PROCESS PROCESSNAME='THR200',PROCESSTYPE='SQR
Report',RUNLOCATION='PSUNX',RUNCONTROLID='mlc',OUTPUTDEST='/tmp/'
```

Das zweite Beispiel verwendet die optionalen Parameter dazu, um die Ausgabe des PeopleSoft-Prozesses als PDF-Datei in ein temporäres Verzeichnis zu speichern.

```
PS_RUN_PROCESS PROCESSNAME='DDDAUDIT',PROCESSTYPE='SQR
Report',RUNLOCATION='PSNT',RUNCONTROLID='sbb',OUTDESTTYPE='FILE',OUTDE
STFORMAT='PDF',OUTPUTDEST='c:\temp'
```

Siehe auch:

Allgemeines zur PeopleSoft-JCL

4.3.7 PS_SET_BINDVAR

Ersetzt den Wert einer Bind-Variablen in einer Prozessdefinition.

Schnittstelle: PROCESSREQUEST_SBB
PeopleSoft-Version: 8+

Syntax

PS_SET_BINDVAR

```
NAME=...
,MODE='V'
,VALUE=...
```

PS_SET_BINDVAR

```
NAME=...
,MODE='D'
,RUNCONTROLID=...
```

Syntaxteil	Beschreibung/Format
NAME=	Name der Bind-Variablen, die ersetzt werden soll Format: Script-Literal
MODE=	Art der Ersetzung Format: Script-Literal Mögliche Werte: "V" und "D" "V" - Der Wert wird mit dem Parameter "VALUE=" vorgegeben. "D" - Die Ersetzung erfolgt über die Angabe einer Run Control ID mit dem Parameter "RUNCONTROLID".
VALUE=	Wert mit dem die Ersetzung vorgenommen werden soll Format: Script-Literal
RUNCONTROLID=	Angabe einer Run Control ID Format: Script-Literal

Anmerkungen

Um das Sprachmittel nutzen zu können, sind die Anweisungen im Abschnitt "Konfiguration zur Nutzung von Bind-Variablen" in der Installationsanleitung des Agenten durchzuführen. In diesem Abschnitt ist das Anlegen der Prozesstypen beschrieben. Anschließend sind die Prozesstypen dem Process Scheduler Server zuzuweisen. Um diese Änderung zu übernehmen, muss der Process Scheduler Server neu gestartet werden.

Beispiele

Die folgenden beiden Beispiele zeigen die Ersetzung einerseits mit einem vorgegebenen Wert und andererseits über eine Run Control ID.

```
PS_SET_BINDVAR NAME=':RUN_CNTL_HR.COURSE',VALUE='K014',MODE='V'
PS_RUN_PROCESS PROCESSNAME='TRN023--
',PROCESSTYPE='Crystal',RUNCONTROLID='ang',RUNLOCATION='PSNT'
```

```
PS_MODIFY_RUNCONTROL RUNCONTROLID='ang',RECORDNAME='RUN_CNTL_
HR',FIELDNAME='COURSE',FIELDVALUE='K014'
PS_SET_BINDVAR NAME=':RUN_CNTL_
HR.COURSE',VALUE='',MODE='D',RUNCONTROLID='ang'
PS_RUN_PROCESS PROCESSNAME='TRN023--
',PROCESSTYPE='Crystal',RUNCONTROLID='ang',RUNLOCATION='PSNT'
```

Siehe auch:

Nutzung von Bind-Variablen
Allgemeines zur PeopleSoft-JCL

4.4 SAP

4.4.1 SAP Basis

Allgemeines zur SAP-JCL

Zeilen mit SAP-JCL können mit :JCL_CONCAT_CHAR verknüpft werden. Für JCL Zeilen gibt es keine Beschränkung.

Einige SAP JCL Sprachmittel sind abhängig von der verwendeten Schnittstelle. Die genauen Syntaxunterschiede sind bei den einzelnen Sprachmitteln beschrieben. Die verfügbaren Sprachmittel je Schnittstelle finden Sie im Kapitel Funktionsunterschiede.

Die Eigenschaften aller Felder, die von AE an die SAP Schnittstellen weitergereicht werden (z.B. Druckername, Programmname) insbesondere maximale Feldlängen, erlaubte Werte, etc., sind NICHT in der Automic Dokumentation enthalten. Die Beschreibung dieser Felder sind je nach SAP Version im SAP Dictionary hinterlegt.

Bei SAP JCL Sprachmittel müssen die Werte der Parameter zwingend nur dann als Script-Literal angegeben werden, wenn der Wert Leerzeichen enthält. Ansonsten ist die Verwendung von Anführungszeichen optional, unabhängig davon ob es sich bei dem Wert um eine Zahl, ein Zeichen oder eine Zeichenkette handelt.

R3_ACTIVATE_CM_PROFILE

Aktiviert ein Profil im SAP Kriterienmanager.

Transaktion: SM62

Schnittstelle: Standard

Syntax

R3_ACTIVATE_CM_PROFILE

ID=...
,**TYPE**=...

Syntaxteil	Beschreibung/Format
ID=	ID des Profils Format: Zahl
TYPE=	Typ des Profils Format: Script-Literal Erlaubte Wert: "EVHIRO", "EVTHIS" und "INTERC" "EVHIRO" - Event History Reorg "EVTHIS" - Event History "INTERC" - Job Interception

Anmerkungen

Pro Profiltyp kann nur ein Profil aktiviert sein.

Beispiel

Das Beispiel aktiviert das Profil mit der ID "5" für die Job Interception.

```
R3_ACTIVATE_CM_PROFILE ID="5",TYPE="INTERC"
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_ACTIVATE_EXT_COMMAND

Ausführen eines externen Kommandos.

Transaktion: SM36

Schnittstelle: Standard

Syntax

R3_ACTIVATE_EXT_COMMAND

COMMAND=...
OPSYSTEM=...
[,**PARAMS**=...]
,**TARGET_SERVER**=...
[,**STDOUT**=...]
[,**STDERR**=...]
[,**TRACE**=...]
[,**TERM**=...]

Syntaxteil	Beschreibung/Format
COMMAND=	Name der Definition des externen Kommandos. Format: Script-Literal
OPSYSTEM=	Hostsystem, in dem das Kommando ausgeführt werden soll. Format: Script-Literal
PARAMS=	Parameter für das externe Kommando. Format: Script-Literal Standardwert: ""
TARGET_S[ERVER]=	Name des Rechners, auf dem das Kommando ausgeführt werden soll. Format: Script-Literal
Steuerflags	
STDOUT=	Externe Ausgabe in Job-Log übernehmen. Format: Script-Literal Erlaubte Werte: "" (Standardwert) und "X" "" - Steuerflag wird nicht gesetzt. "X" - Externe Ausgabe wird ins Job-Log übernommen.
STDERR=	Externe Fehler in Job-Log übernehmen. Format: Script-Literal Erlaubte Werte: "" (Standardwert) und "X" "" - Steuerflag wird nicht gesetzt. "X" - Externe Fehler werden ins Job-Log übernommen.
TRACE=	Trace aktivieren Format: Script-Literal Erlaubte Werte: "" (Standardwert) und "X" "" - Steuerflag wird nicht gesetzt. "X" - Trace wird aktiviert.
TERM=	Job wartet auf externe Beendigung Format: Script-Literal Erlaubte Werte: "" (Standardwert) und "X" "" - Steuerflag wird nicht gesetzt. "X" - Job wartet auf externe Beendigung.

Beispiel

```
R3_ACTIVATE_EXT_COMMAND COMMAND='NET_ROUTING',OPSYSTEM='Windows
NT',TARGET_SERVER='NB0053',STDOUT='X',STDERR='X',TERM='X'
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_ACTIVATE_EXT_PROGRAM

Ausführen eines externen Programmes.

Transaktion: SM36

Schnittstelle: Standard

Syntax

R3_ACTIVATE_EXT_PROGRAM

PROGRAM=...
[,**PARAMS**=...]
,**TARGET_SERVER**=...
[,**STDOUT**=...]
[,**STDERR**=...]
[,**TRACE**=...]
[,**TERM**=...]

Syntaxteil	Beschreibung/Format
PROGRAM=	Name des auszuführenden Programmes Format: Script-Literal
PARAMS=	Parameter für das Programm Format: Script-Literal Standardwert: ""
TARGET_S[ERVER]=	Name des Rechners, auf dem das Programm ausgeführt werden soll. Format: Script-Literal
Steuerflags	
STDOUT=	Externe Ausgabe in Job-Log übernehmen Format: Script-Literal Erlaubte Werte: "" (Standardwert) und "X" "" - Steuerflag wird nicht gesetzt. "X" - Externe Ausgabe wird ins Job-Log übernommen.
STDERR=	Externe Fehler in Job-Log übernehmen Format: Script-Literal Erlaubte Werte: "" (Standardwert) und "X" "" - Steuerflag wird nicht gesetzt. "X" - Externe Fehler werden ins Job-Log übernommen.
TRACE=	Trace aktivieren Format: Script-Literal Erlaubte Werte: "" (Standardwert) und "X" "" - Steuerflag wird nicht gesetzt. "X" - Trace wird aktiviert.

TERM=	Job wartet auf externe Beendigung Format: Script-Literal Erlaubte Werte: "" (Standardwert) und "X" "" - Steuerflag wird nicht gesetzt. "X" - Job wartet auf externe Beendigung.

Beispiel

```
R3_ACTIVATE_EXT_PROGRAM PROGRAM='dir',PARAMS='*.*',TARGET_
SERVER='NB0053',STDOUT='X',STDERR='X',TERM='X'
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_ACTIVATE_INTERCEPTED_JOBS

Führt Intercepted Jobs unter der Kontrolle von AE durch. Die zu startenden Intercepted Jobs werden über eine Selektion ermittelt.

Transaktion: SM37

Schnittstelle: Standard (XBP 2.0)

Syntax

R3_ACTIVATE_INTERC[EPTED_JOBS]

NAME=...
[**,GROUP=**...]
[**,USER=**...]
[**,NOFOUND=**...]
[**,ERROR=**...]
[**,SELECT=**...]
[**,START_DATE=**...]
[**,START_TIME=**...]
[**,END_DATE=**...]
[**,END_TIME=**...]
[**,JOBCOUNT=**...]
[**,TARGET_SERVER=**...]
[**,WAIT=**...]
[**,REPLICATE**=...]
[**,ABORTED=**...]
[**,GET_SPOOL=...**]

Syntaxteil	Beschreibung/Format

NAME=	Selektion eines oder mehrerer Intercepted Jobs nach Namen Format des Wertes: Script-Literal Name und Nummer identifizieren einen einzelnen SAP-Job eindeutig. Bei der Selektion mehrerer Jobs kann das Wildcardzeichen "*" verwendet werden (z.B. "xxx*").
GROUP=	Selektion von Intercepted Jobs nach Gruppen (z.B. "xxx*") Format des Wertes: Script-Literal Standardwert: "*"
USER=	Selektion von Intercepted Jobs nach Benutzern (z.B. "xxx*") Format des Wertes: Script-Literal Standardwert: "*"
NOFOUND=	Behandlung, wenn durch die Selektion keine Intercepted Jobs gefunden werden Format des Wertes: Script-Literal Erlaubte Werte: "NORMAL" (Standardwert) und "ABEND" "NORMAL" - Das Script wird fortgesetzt, der AE Job endet normal. "ABEND" - Das Script wird nicht fortgesetzt, der AE Job endet abnormal.
ERROR=	Behandlung, wenn ein selektierter Intercepted Job abnormal endet Format des Wertes: Script-Literal Erlaubte Werte: "ABEND" (Standardwert) und "IGNORE" "ABEND" - Das Script wird nicht fortgesetzt, der AE Job endet abnormal. "IGNORE" - Das Script wird fortgesetzt, der AE Job endet normal.
SEL[ECT]=	Einmalig oder permanent selektieren Format des Wertes: Script-Literal Erlaubte Werte: "ONCE" (Standardwert) und "EVERY" "ONCE" - Es wird die Treffermenge (= Anzahl eingeplanter Jobs) einmalig ermittelt. "EVERY" - Nach jeder Freigabe eines Jobs wird die Treffermenge erneut ermittelt. Diese Technik erlaubt das Parallelisieren eingeplanter Jobs.
START_D [ATE]=	Start-Datum der Selektion Format des Wertes: Script-Literal Datumsformat: JJJJMMTT Standardwert: "20000101" Wird dieser Parameter ohne den Parameter END_DATE= verwendet, werden alle eingeplanten Jobs ab dem angegebenen Start-Datum selektiert und gestartet.
START_T [IME]=	Start-Zeit der Selektion Format des Wertes: Script-Literal Zeitformat: HHMMSS Standardwert: "000000" Wird dieser Parameter ohne den Parameter END_TIME= verwendet, werden alle eingeplanten Jobs ab der angegebenen Start-Zeit selektiert und gestartet. 23:59 Uhr gilt als Ende-Zeit.

END_D[ATE]=	Ende-Datum der Selektion Format des Wertes: Script-Literal Datumsformat: JJJJMMTT Standardwert: aktuelles Datum Um alle eingeplanten Jobs eines Tages zu selektieren, muss dieser Parameter angegeben und das gleiche Datum wie beim Parameter START_DATE= verwendet werden.
END_T[IME]=	Ende-Zeit der Selektion Format des Wertes: Script-Literal Zeitformat: HHMMSS Standardwert: "235959"
JOBCOUNT=	Nummer des Intercepted Jobs Format des Wertes: Script-Literal Standardwert: "*" Name und Nummer identifizieren einen einzelnen SAP-Job eindeutig.
TARGET_S[ERVER]=	Zielsystem, welches ein Intercepted Job verwenden soll Format des Wertes: Script-Literal Erlaubte Werte: "KEEP" (Standardwert) und "ATTRIBUTE" "KEEP" - Das im Originaljob eingetragene Zielsystem wird beibehalten. "ATTRIBUTE" - Das Zielsystem aus den Host-Attributen des Jobs wird verwendet.
WAIT=	Auf das Ende der Children eines Intercepted Jobs warten Format des Wertes: Script-Literal Erlaubte Werte: "YES" und "NO" (Standardwert) "NO" - Es wird nicht auf das Ende aller Children gewartet. "YES" - Es wird auf das Ende aller Children gewartet.
REPL[ICATE]=	Behandlung der Children von Intercepted Jobs Format des Wertes: Script-Literal Erlaubte Werte: "YES" und "NO" (Standardwert) "YES" - Children eines Jobs werden im Automation Engine System repliziert. Sie erscheinen sodann im Aktivitätenfenster des UserInterfaces. Des Weiteren werden Statistiksätze und Reports im Automation Engine System erzeugt. "NO" - Es erfolgt keine Nachbildung im Automation Engine System.
ABORTED=	Auf das abnormale Ende der Children eines Intercepted Jobs reagieren Format des Wertes: Script-Literal Erlaubte Werte: "YES" (Standardwert) und "NO" "YES" = Das Script wird nicht fortgesetzt, der AE Job endet abnormal. "NO" = Das Script wird fortgesetzt, der AE Job endet normal.

GET_ SPOOL=	Spoolliste des gestarteten Jobs anfordern Format des Wertes: Script-Literal Erlaubte Werte: "YES" oder "NO" (Standardwert) "YES" = Spoolliste anfordern. Diese wird dabei als Text-Datei in das Verzeichnis gespeichert, welches Sie in der INI-Datei des SAP-Agenten mit dem Parameter Download_dir= (Sektion [GLOBAL])festlegen. Der Name dieser Datei setzt sich folgendermaßen zusammen: Zusätzlich wird diese Datei beim AE-Job als Job-Output registriert. "NO" = Spooliste wird nicht angefordert

Anmerkungen

Das Script-Sprachmittel ermittelt alle Intercepted Jobs nach den angegebenen Selektionskriterien. Bei der Selektion wird der SAP-Mandant verwendet, der im Login-Objekt, auf welches der AE Job referenziert, angegeben ist.

Beispiel

Im Beispiel sollen alle Intercepted Jobs gestartet werden. Werden keine Intercepted Jobs gefunden, soll das nicht zum Abbruch des AE Jobs führen.

```
R3_ACTIVATE_INTERCEPTED_JOBS
NAME='*',GROUP='*',USER='*',NOFOUND='NORMAL'
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_ACTIVATE_JOBS

Führt bereits im SAP eingeplante Jobs unter der Kontrolle von AE durch. Die zu startenden Jobs werden über eine Selektion ermittelt.

Transaktion: SM37

Schnittstelle: AE und Standard

Automation Engine Schnittstelle

[AE] [Standard]

Syntax

R3_ACTIVATE_JOB[S]

NAME=...
[**,JOBCOUNT=**...]
[**,GROUP=**...]
[**,USER=**...]
[**,NOFOUND=**...]

[**,ERROR=**...]
[**,SELECT=**...]
[**,START=**...]
[**,NEW_NAME=**...]
[**,START_D[ATE]=**...]
[**,START_T[IME]=**...]
[**,END_D[ATE]=**...]
[**,END_T[IME]=**...]
[**,NO_DATE=**...]
[**,TARGET_S[ERVER]=**...]
[**,BEG_LOGLINES=**...]
[**,END_LOGLINES=**...]
[**,GET_SPOOL=**...]

Syntaxteil	Beschreibung/Format
NAME=	Selektion eines oder mehrerer Jobs nach Namen Format: Script-Literal oder AE-name Name und Nummer identifizieren einen einzelnen SAP-Job eindeutig. Bei der Selektion mehrerer Jobs kann das Wildcardzeichen "*" verwendet werden (z.B. "xxx*").
Parameter	Weitere Selektionskriterien in Form eines Schlüsselwortes und einer Zuweisung. Nicht angegebene Parameter erhalten Standardwerte. Formate gelten für die Zuweisung zum Schlüsselwort.
JOBCOUNT=	Nummer des SAP-Jobs Format: Script-Literal Name und Nummer identifizieren einen einzelnen SAP-Job eindeutig.
GROUP=	Selektion von Jobs nach Gruppen (z.B. "xxx*") Format: Script-Literal oder AE-Name Standardwert: "*"
USER=	Selektion von Jobs nach Benutzern (z.B. "xxx*") Format: Script-Literal oder AE-Name Standardwert: "*"
NOFOUND=	Behandlung, wenn durch die Selektion keine Jobs getroffen wurden Format: Script-Literal Erlaubte Werte: "NORMAL" (Standardwert) und "ABEND" "NORMAL" - Das Script wird fortgesetzt, der AE Job endet normal. "ABEND" - Das Script wird nicht fortgesetzt, der AE Job endet abnormal.

Syntaxteil	Beschreibung/Format
ERROR=	Behandlung wenn ein Job aus der selektieren Menge abnormal endet Format: Script-Literal Erlaubte Werte: "ABEND" (Standardwert) und "IGNORE" "ABEND" - Das Script wird nicht fortgesetzt, der AE Job endet abnormal "IGNORE" - Das Script wird fortgesetzt, der AE Job endet normal.
SELECT=	Einmalig oder permanent selektieren Format: Script-Literal Erlaubte Werte "ONCE" (Standardwert) und "EVERY" "ONCE" - Es wird die Treffermenge (= Anzahl freizugebender Jobs) einmalig ermittelt. "EVERY" - Nach jeder Freigabe eines Jobs wird die Treffermenge erneut ermittelt. Diese Technik erlaubt das Parallelisieren freigegebener Jobs.
START=	Soll das Original des Jobs oder ein Duplikat gestartet werden Format: Script-Literal Erlaubte Werte: "ORIGINAL" (Standardwert) und "DUPLICATE"
NEW_NAME=	Name für den Job, der als Duplikat gestartet wird Format: Script-Literal Wird nur ausgewertet, wenn START=DUPLICATE angegeben wurde. Soll das Original des Jobs gestartet werden, wird der Parameter ignoriert.
START_D[ATE]=	Start-Datum der Selektion von eingeplanten Jobs Format: Script-Literal Datumsformat: JJJJMMTT Standardwert: "20000101" Wird dieser Parameter ohne den Parameter END_DATE verwendet, werden alle eingeplanten Jobs ab dem angegebenen Start-Datum selektiert und gestartet.
START_T[IME]=	Start-Zeit der Selektion von eingeplanten Jobs Format: Script-Literal Zeitformat: HHMMSS Standardwert: "000000" Wird dieser Parameter ohne den Parameter END_TIME verwendet, werden alle eingeplanten Jobs ab der angegebenen Start-Zeit selektiert und gestartet. 23:59 Uhr gilt als Ende-Zeit.

Syntaxteil	Beschreibung/Format
END_D[ATE]=	Ende-Datum der Selektion von eingeplanten Jobs Format: Script-Literal Datumsformat: JJJJMMTT Standardwert: aktuelles Datum Um alle eingeplanten Jobs eines Tages zu selektieren, muss dieser Parameter angegeben und das gleiche Datum wie beim Parameter START_DATE verwendet werden.
END_T[IME]=	Ende-Datum der Selektion von eingeplanten Jobs Format: Script-Literal Datumsformat: JJJJMMTT Standardwert: "235959" Um alle eingeplanten Jobs eines Tages zu selektieren, muss dieser Parameter angegeben und das gleiche Datum wie beim Parameter START_DATE verwendet werden.
NO_DATE=	Jobs ohne Starttermin Format: Script-Literal oder AE-Name Erlaubte Werte: "" (Standardwert) oder "X"
TARGET_S[ERVER]=	Zielsystem, das ein freigegebener SAP-Job verwenden soll Format: Script-Literal Erlaubte Werte: "KEEP" (Standardwert) und "ATTRIBUTE" "KEEP" - Das im Originaljob eingetragene Zielsystem wird beibehalten. "ATTRIBUTE" - Das Zielsystem aus den Host-Attributen des Jobs wird verwendet.
BEG_LOGLINES=	Legt fest wie viele Zeilen vom Beginn des SAP Joblogs in den AE-Jobreport übernommen werden sollen.
END_LOGLINES=	Anzahl der Zeilen, ab Ende des SAP Joblogs, die in den AE-Jobreport geschrieben werden. Wird weder der Parameter END_LOGLINES noch BEG_ LOGLINES angegeben, so wird standardmäßig das komplette Joblog verwendet. Wenn nur ein Parameter angeführt wird, so werden entweder nur Zeilen vom Beginn oder Ende des Joblogs gelesen. Wenn "0" als Wert für beide Parameter eingetragen ist, wird kein Joblog übertragen. Für beide Parameter sind nur numerische Werte erlaubt.

Syntaxteil	Beschreibung/Format
GET_SPOOL=	Spoolliste des gestarteten Jobs anfordern Format des Wertes: Script-Literal Erlaubte Werte: "YES" oder "NO" (Standardwert) "YES" = Spoolliste anfordern. Diese wird dabei als Text-Datei in das Verzeichnis gespeichert, welches Sie in der INI-Datei des SAP-Agenten mit dem Parameter Download_dir= (Sektion [GLOBAL])festlegen. Der Name dieser Datei setzt sich folgendermaßen zusammen: <*SAP-Job-Count*>_<*Step-Nummer*>_<*Spool-Nummer*>.txt Zusätzlich wird diese Datei beim AE-Job als Job-Output registriert. "NO" = Spooliste wird nicht angefordert

Beispiel

Alle eingeplanten Jobs des SAP-Benutzers 'NI' freigeben.

```
R3_ACTIVATE_JOBS NAME='*',GROUP='*',USER='NI'
```

Im zweiten Beispiel werden alle Jobs, die zwischen dem 1.7.2001 00:00:00 Uhr und dem 2.7.2001 23:59:59 Uhr eingeplant sind, selektiert und gestartet.

```
R3_ACTIVATE_JOBS NAME='*',GROUP='*',USER='SUPPORT',START_
DATE='20010701',END_DATE='20010702'
```

Im Beispiel werden alle Jobs, die zwischen dem 1.7.2001 12:30:00 Uhr und dem 2.7.2001 12:30:00 Uhr eingeplant sind, selektiert und gestartet.

```
R3_ACTIVATE_JOBS NAME='*',GROUP='*',USER='SUPPORT',START_
DATE='20010701', START_TIME='123000',END_DATE='20010702',END_
TIME='123000'
```

Standard Schnittstelle

[AE] [Standard]

Syntax

R3_ACTIVATE_JOB[S]

NAME=...
[**,JOBCOUNT=**...]
[**,NOFOUND=**...]
[**,ERROR=**...]
[**,SELECT=**...]
[**,START=**...]
[**,NEW_NAME=**...]
[**,START_D[ATE]=**...]
[**,START_T[IME]=**...]
[**,END_D[ATE]=**...]

[**,END_T[IME]=**...]
[**,TARGET_S[ERVER]=**...]
[**,MON[ITOR]=**...]
[**,WAIT=**...]
[**,ABORTED=**...]
[**,REPL[ICATE]=**...]
[**,BEG_LOGLINES=**...]
[**,END_LOGLINES=**...]
[**,GET_SPOOL=...**]

Syntaxteil	Beschreibung/Format
NAME=	Selektion eines oder mehrerer Jobs nach Namen Format: Script-Literal oder AE-name Name und Nummer identifizieren einen einzelnen SAP-Job eindeutig. Bei der Selektion mehrerer Jobs kann das Wildcardzeichen "*" verwendet werden (z.B. "xxx*").
Parameter	Weitere Selektionskriterien in Form eines Schlüsselwortes und einer Zuweisung. Nicht angegebene Parameter erhalten Standardwerte. Formate gelten für die Zuweisung zum Schlüsselwort.
JOBCOUNT=	Nummer des SAP-Jobs Format: Script-Literal Name und Nummer identifizieren einen einzelnen SAP-Job eindeutig.
NOFOUND=	Behandlung, wenn durch die Selektion keine Jobs getroffen wurden Format: Script-Literal Erlaubte Werte: "NORMAL" (Standardwert) und "ABEND" "NORMAL" - Das Script wird fortgesetzt, der AE Job endet normal. "ABEND" - Das Script wird nicht fortgesetzt, der AE Job endet abnormal.
ERROR=	Behandlung wenn ein Job aus der selektieren Menge abnormal endet Format: Script-Literal Erlaubte Werte: "ABEND" (Standardwert) und "IGNORE" "ABEND" - Das Script wird nicht fortgesetzt, der AE Job endet abnormal "IGNORE" - Das Script wird fortgesetzt, der AE Job endet normal.

Syntaxteil	Beschreibung/Format
SELECT=	Einmalig oder permanent selektieren Format: Script-Literal Erlaubte Werte "ONCE" (Standardwert) und "EVERY" "ONCE" - Es wird die Treffermenge (= Anzahl freizugebender Jobs) einmalig ermittelt. "EVERY" - Nach jeder Freigabe eines Jobs wird die Treffermenge erneut ermittelt. Diese Technik erlaubt das Parallelisieren freigegebener Jobs.
START=	Soll das Original des Jobs oder ein Duplikat gestartet werden Format: Script-Literal Erlaubte Werte: "ORIGINAL" (Standardwert) und "DUPLICATE"
NEW_NAME=	Name für den Job, der als Duplikat gestartet wird Format: Script-Literal Wird nur ausgewertet, wenn START=DUPLICATE angegeben wurde. Soll das Original des Jobs gestartet werden, wird der Parameter ignoriert.
START_D[ATE]=	Start-Datum der Selektion von eingeplanten Jobs Format: Script-Literal Datumsformat: JJJJMMTT Standardwert: "20000101" Wird dieser Parameter ohne den Parameter END_DATE verwendet, werden alle eingeplanten Jobs ab dem angegebenen Start-Datum selektiert und gestartet.
START_T[IME]=	Start-Zeit der Selektion von eingeplanten Jobs Format: Script-Literal Zeitformat: HHMMSS Standardwert: "000000" Wird dieser Parameter ohne den Parameter END_TIME verwendet, werden alle eingeplanten Jobs ab der angegebenen Start-Zeit selektiert und gestartet. 23:59 Uhr gilt als Ende-Zeit.
END_D[ATE]=	Ende-Datum der Selektion von eingeplanten Jobs Format: Script-Literal Datumsformat: JJJJMMTT Standardwert: aktuelles Datum Um alle eingeplanten Jobs eines Tages zu selektieren, muss dieser Parameter angegeben und das gleiche Datum wie beim Parameter START_DATE verwendet werden.

Syntaxteil	Beschreibung/Format
END_T[IME]=	Ende-Datum der Selektion von eingeplanten Jobs Format: Script-Literal Datumsformat: JJJJMMTT Standardwert: "235959" Um alle eingeplanten Jobs eines Tages zu selektieren, muss dieser Parameter angegeben und das gleiche Datum wie beim Parameter START_DATE verwendet werden.
TARGET_S[ERVER]=	Zielsystem, das ein freigegebener SAP-Job verwenden soll Format: Script-Literal Erlaubte Werte: "KEEP" (Standardwert) und "ATTRIBUTE" "KEEP" - Das im Originaljob eingetragene Zielsystem wird beibehalten. "ATTRIBUTE" - Das Zielsystem aus den Host-Attributen des Jobs wird verwendet.
MON[ITOR]=	Statusüberwachung im Aktivierungslog protokollieren Format: Script-Literal Erlaubte Werte: "YES" und "NO" (Standardwert)
WAIT=	Auf das Ende der Children eines SAP-Jobs warten Format des Wertes: Script-Literal oder AE-Name Erlaubte Werte: "YES" und "NO" (Standardwert) "YES" - Es wird auf das Ende aller Children gewartet. Children werden im Aktivierungsprotokoll protokolliert. "NO" - Es wird nicht auf das Ende aller Children gewartet.
ABORTED=	Auf das abnormale Ende der Children eines SAP-Jobs reagieren Format des Wertes: Script-Literal oder AE-Name Erlaubte Werte: "YES" (Standardwert) und "NO" "YES" - SAP-Job (Parent) wird abgebrochen. Das Script wird nicht fortgesetzt und der AE Job endet abnormal. "NO" - SAP-Job (Parent) wird nicht abgebrochen. Das Script wird fortgesetzt und der AE Job endet normal.
REPL[ICATE]=	Behandlung der Children von Jobs Format des Wertes: Script-Literal Erlaubte Werte: "YES" und "NO" (Standardwert) "YES" - Children eines Jobs werden im Automation Engine System repliziert. Sie erscheinen im Aktivitätenfenster des UserInterfaces. Des Weiteren werden Statistiksätze und Reports im Automation Engine System erzeugt. "NO" - Es erfolgt keine Nachbildung im Automation Engine System.
BEG_LOGLINES=	Legt fest wie viele Zeilen vom Beginn des SAP Joblogs in den AE-Jobreport übernommen werden sollen.

Syntaxteil	Beschreibung/Format
END_LOGLINES=	Anzahl der Zeilen, ab Ende des SAP Joblogs, die in den AE-Jobreport geschrieben werden. Wird weder der Parameter END_LOGLINES noch BEG_ LOGLINES angegeben, so wird standardmäßig das komplette Joblog verwendet. Wenn nur ein Parameter angeführt wird, so werden entweder nur Zeilen vom Beginn oder Ende des Joblogs gelesen. Wenn "0" als Wert für beide Parameter eingetragen ist, wird kein Joblog übertragen. Für beide Parameter sind nur numerische Werte erlaubt.
GET_SPOOL=	Spoolliste des gestarteten Jobs anfordern Format des Wertes: Script-Literal Erlaubte Werte: "YES" oder "NO" (Standardwert) "YES" = Spoolliste anfordern. Diese wird dabei als Text-Datei in das Verzeichnis gespeichert, welches Sie in der INI-Datei des SAP-Agenten mit dem Parameter Download_dir= (Sektion [GLOBAL])festlegen. Der Name dieser Datei setzt sich folgendermaßen zusammen: Zusätzlich wird diese Datei beim AE-Job als Job-Output registriert. "NO" = Spooliste wird nicht angefordert

Beispiel

Alle eingeplanten Jobs mit Jobnamen 'REORG_SPOOL' freigeben.

```
R3_ACTIVATE_JOBS NAME='REORG_SPOOL',NOFOUND='ABEND'
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_ACTIVATE_REPORT

Führt den angegebenen Report aus. Wenn notwendig, können eine Variante und diverse Parameter zur Liststeuerung angegeben werden.

Transaktion: SM36, SM37

Schnittstelle: AE und Standard

Syntax

R3_ACTIVATE_REP[ORT]

REPORT=...
[**,VARIANT=**...]
[**,DESTINATION=**...]

[**,COPIES=**...]
[**,LIST_NAME=**...]
[**,LIST_TEXT=**...]
[**,IMMEDIATELY=**...]
[**,RELEASE=**...]
[**,NEW_LIST_ID=**...]
[**,EXPIRATION=**...]
[**,LINE_COUNT=**...]
[**,LINE_SIZE=**...]
[**,LAYOUT=**...]
[**,COVERPAGE=**...]
[**,SAP_COVER_PAGE=**...]
[**,OS_COVER_PAGE=**...]
[**,RECEIVER=**...]
[**,DEPARTMENT=**...]
[**,AUTHORITY=**...]
[**,DATA_SET=**...]
[**,TYPE=**...]
[**,SPOOL_PRIORITY=**...]
[**,TEXTONLY=**...]
[**,FRAMES=**...]
[,**SUPPRESS_SHADING=**...]
[,**WITH_STRUCTURE=**...]
[,**DEFAULT_SPOOL_SIZE=**...]
[,**PRINTER_MAIL_ADDRESS=**...]
[,**SPOOL_PAGE_FROM=**...]
[,**SPOOL_PAGE_TO=**...]
[**,ARCHIVE_MODE=**...]
[**,ARCHIVE_SAPOBJECT=**...]
[**,ARCHIVE_OBJECT=**...]
[**,ARCHIVE_INFO=**...]
[**,ARCHIVE_TEXT=**...]
[**,MONITOR=**...]
[**,WAIT=**...]
[**,ABORTED=**...]
[**,REPL[ICATE]=**...]
[**,BEG_LOGLINES=**...]
[**,END_LOGLINES=**...]
[**,GET_SPOOL=...**]

Syntaxteil	Beschreibung/Format
REP[ORT]=	Name des Reports Format: Script-Literal
VAR[IANT]=	Name der Variante Format: Script-Literal
Druckparameter	

Syntaxteil	Beschreibung/Format
DEST[INATION]=	Ausgabegerät Format: Script-Literal Mit diesem Parameter geben Sie den Namen des Ausgabegerätes an. Dies ist in den meisten Fällen ein Druckername, kann aber auch der Name eines Telefaxgerätes o.ä. sein. ⚠ Beachten Sie, dass die Nutzung der langen SAP-Druckernamen innerhalb von AE aus technischen Gründen, die an der XBP-Schnittstelle von SAP liegen, nicht möglich ist. Geben Sie daher den 4-stelligen technischen Namen an.
COPIES=	Anzahl Ausdrucke Format: Integer Standardwert: "0" Geben Sie hier an, wieviel Exemplare Ihres Dokumentes ausgedruckt werden sollen.
LIST_N[AME]=	Name des Spool-Auftrags Format: Script-Literal Mit diesem Parameter geben Sie den Namen des Spool-Auftrages an. Dieser darf beliebig aus Buchstaben, Ziffern, Sonderzeichen und Leerzeichen bestehen. Der Standardname, der für einen Spool-Auftrag vorgeschlagen wird, besteht aus den 8 Zeichen des Reportnamens, dem Trennzeichen '_' und den ersten 3 Zeichen des Benutzernamens.
LIST_T[EXT]=	Text für Deckblatt Format: Script-Literal Mit diesem Parameter geben Sie den Beschreibungstext zum Spool-Auftrag an. Dieser darf beliebig aus Buchstaben, Ziffern, Sonderzeichen und Leerzeichen bestehen.
IMM[EDIATELY]=	Sofort ausgeben Format: Script-Literal Erlaubte Werte: "YES" und "NO" (Standardwert)
REL[EASE]=	Löschen nach Ausgabe Format: Script-Literal Erlaubte Werte: "YES" und "NO" (Standardwert) Mit diesem Parameter wird bestimmt, ob der Spool-Auftrag sofort nach Abschluss der Ausgabe zum Ausgabegerät oder erst nach Ablauf der Spool-Verweildauer gelöscht wird.
NEW_LIST_ID=	Neuer Spool-Auftrag Format: Script-Literal Erlaubte Werte: "YES" (Standardwert) oder "NO"

Syntaxteil	Beschreibung/Format
EXPIR[ATION]=	Spool-Verweildauer Format: Integer Standardwert: "0" Dieser Parameter bestimmt, wieviel Tage ein Spool-Auftrag im Spool-System verweilen soll, bevor er gelöscht wird.
LINE_COUNT=	Seitenlänge der Liste Format: Integer Standardwert: "0" Zeilenzahl pro Seite einer Liste. Wenn dieses Feld eine Null enthält oder leer ist, ist die Seitenzahl für die Liste unbegrenzt (beim Drucken nicht erlaubt). Die Länge der Liste wird dann allein durch ihren Inhalt bestimmt. Beim Drucken hängt die maximale Zeilenzahl pro Seite von der gewählten Aufbereitung ab. Wollen Sie die Zeilenzahl ändern, dann müssen Sie eine andere Aufbereitung wählen.
LINE_SIZE=	Zeilenbreite der Liste Format: Integer Standardwert: "0" Dieser Parameter enthält die aktuelle Zeilenbreite der Liste. Beim Drucken hängt die maximale Zeilenbreite von der gewählten Aufbereitung ab. Wollen Sie die Zeilenbreite ändern, so müssen Sie eine andere Aufbereitung wählen.
LAYOUT=	Druckaufbereitung Format: Script-Literal Dieser Parameter bestimmt die Aufbereitung des Spool-Auftrags bei der Ausgabe. Durch die Aufbereitung wird im wesentlichen das Seitenformat definiert, d.h. die maximale Anzahl der Zeilen und Spalten auf einer Druckseite.
COVER[PAGE]=	Selektions-Deckblatt Format: Script-Literal Erlaubte Werte: "YES" oder "NO" (Standardwert) Dieser Parameter bestimmt, ob am Beginn der Liste ein Deckblatt mit den Reportselektionen ausgegeben werden soll. Wenn ein Deckblatt erzeugt wird, so wird dieses auch in den Jobreport übernommen. Damit ist dokumentiert, mit welchen Parametern diese Durchführung erfolgte.

Syntaxteil	Beschreibung/Format
SAP_COVER[_PAGE]=	SAP-Deckblatt Format: Script-Literal Erlaubte Werte: "" (Standardwert), "X" und "D" "" - Kein Deckblatt ausgeben "X" - Deckblatt ausgeben "D" - Deckblattausgabe hängt von der Einstellung des jeweiligen Ausgabegerätes (Drucker) ab Dieser Parameter bestimmt, ob bei Ausgabe des Spool-Auftrags ein Deckblatt mit verschiedenen Angaben wie Empfängernamen, Abteilungsname, verwendete Aufbereitung u.ä. ausgegeben werden soll.
OS_COVER[_PAGE]=	Hostspooler-Deckblatt Format: Script-Literal Erlaubte Werte: "" (Standardwert), "X" und "D" "" - Kein Deckblatt ausgeben "X" - Deckblatt ausgeben "D" - Deckblattausgabe hängt von der Einstellung des jeweiligen Ausgabegerätes (Drucker) ab
RECEIVER=	Empfänger Format: Script-Literal Dieser Parameter enthält den Namen des Empfängers des Spool-Auftrages. Dieser Name wird beim Ausdruck auf dem Deckblatt ausgegeben. Der Standardwert für den Empfängernamen ist der aktuelle Benutzername.
DEPART[MENT]=	Abteilung im Deckblatt Format: Script-Literal Dieser Parameter enthält den Namen der Abteilung zum Spool-Auftrag. Dieser Name wird beim Ausdruck auf dem Deckblatt ausgegeben.
AUTHORITY=	Berechtigung Format: Script-Literal Dieser Parameter enthält die Berechtigung für den Spool-Auftrag (maximal 12 Stellen). Es können sich nur Benutzer den Inhalt des Spool-Auftrags anzeigen lassen, welche die angegebene Berechtigung besitzen.
DATA_SET=	Name des Spool-Datasets Format: Script-Literal
TYPE=	Name des Spool-Datasets Format: Script-Literal
SPOOL_PRI[ORITY]=	Priorität eines Spoolauftrages Format: Integer Standardwert: "5"

Syntaxteil	Beschreibung/Format
TEXTO[NLY]=	Nur Text Format: Script-Literal Erlaubte Werte: "YES", "NO" (Standardwert) Steuert die Ausgabe der Nicht-ASCII-Zeichen einer Druckliste.
FRAMES=	Rahmenzeichen Format: Script-Literal Erlaubte Werte: "YES" (Standardwert) und "NO" Steuert die standardmäßige Rahmenbildung.
SUPPRESS_SHADING=	Farben und Schatten nicht drucken. Erlaubte Werte: "YES", "NO" (Standardwert)
WITH_STRUCTURE=	Strukturierte Informationen einbeziehen. Erlaubte Werte: "YES", "NO" (Standardwert)
DEFAULT_SPOOL_SIZE=	255 Zeichen als maximale Zeilenbreite definieren. Erlaubte Werte: "YES", "NO" (Standardwert) "YES" = nach der 255. Spalte wird ein Zeilenumbruch erzwungen "NO" = keine Beschränkung der Zeilenlänge
PRINTER_MAIL_ADDRESS=	E-Mail Adresse eines E-Mail Druckers Format: Script-Literal
SPOOL_PAGE_FROM= **SPOOL_PAGE_TO=**	Nummer der Seite, ab und/oder bis zu welcher gedruckt werden soll. Format: Zahl Standardmässig werden alle Seiten gedruckt.
Archivierungsparameter	
ARCHIVE_M[ODE]=	Ablagemodus Format: Script-Literal Erlaubte Werte: "1" (Standardwert), "2" und "3" "1" - Dokument wird nur gedruckt. "2" - Dokument wird nur im optischen Archiv gespeichert. "3" - Dokument wird sowohl gedruckt als auch im Archiv gespeichert.
ARCHIVE_S[APOBJECT]=	Objekttyp des Business Objekts Format: Script-Literal Mittels Objekttypen werden SAP-Objekte klassifiziert. Siehe:Archiv-Parameter bei R3_ACTIVATE_REPORT
ARCHIVE_O[BJECT]=	Dokumentart Format: Script-Literal Archivobjekte werden nach Dokumentarten klassifiziert.

Syntaxteil	Beschreibung/Format
ARCHIVE_I[NFO]=	Infofeld Format: Script-Literal Informationskürzel zum Archivierungsauftrag.
ARCHIVE_T[EXT]=	Textinformationsfeld Format: Script-Literal Beschreibungstext zum Archivierungsauftrag. Dieser darf beliebig aus Buchstaben, Ziffern, Sonderzeichen und Leerzeichen bestehen.
Steuernde Parameter	
MON[ITOR]=	Statusüberwachung im Aktivierungslog protokollieren Format: Script-Literal Erlaubte Werte: "YES" und "NO" (Standardwert)
Parameter für Parent-Child-Beziehung	
WAIT=	Auf das Ende der Children eines SAP-Jobs warten Format des Wertes: Script-Literal Erlaubte Werte: "YES" und "NO" (Standardwert) "YES" - Es wird auf das Ende aller Children gewartet. Children werden im Aktivierungsprotokoll protokolliert. "NO" - Es wird nicht auf das Ende aller Children gewartet.
ABORTED=	Auf das abnormale Ende der Children eines SAP-Jobs reagieren Format des Wertes: Script-Literal Erlaubte Werte: "YES" (Standardwert) und "NO" "YES" - SAP-Job (Parent) wird abgebrochen. Das Script wird nicht fortgesetzt und der AE Job endet abnormal. "NO" - SAP-Job (Parent) wird nicht abgebrochen. Das Script wird fortgesetzt und der AE Job endet normal.
REPL[ICATE]=	Behandlung der Children von Jobs Format des Wertes: Script-Literal Erlaubte Werte: "YES" und "NO" (Standardwert) "YES" - Children eines Jobs werden im Automation Engine System repliziert. Sie erscheinen sodann im Aktivitätenfenster des UserInterfaces. Des Weiteren werden Statistiksätze und Reports im Automation Engine System erzeugt. "NO" - Es erfolgt keine Nachbildung im Automation Engine System.
BEG_LOGLINES=	Legt fest wie viele Zeilen vom Beginn des SAP Joblogs in den AE-Jobreport übernommen werden sollen.

Syntaxteil	Beschreibung/Format
END_LOGLINES=	Anzahl der Zeilen, ab Ende des SAP Joblogs, die in den AE-Jobreport geschrieben werden. Wird weder der Parameter END_LOGLINES noch BEG_LOGLINES angegeben, so wird standardmäßig das komplette Joblog verwendet. Wenn nur ein Parameter angeführt wird, so werden entweder nur Zeilen vom Beginn oder Ende des Joblogs gelesen. Wenn "0" als Wert für beide Parameter eingetragen ist, wird kein Joblog übertragen. Für beide Parameter sind nur numerische Werte erlaubt.
GET_SPOOL=	Spoolliste des gestarteten Reports anfordern Format des Wertes: Script-Literal Erlaubte Werte: "YES" oder "NO" (Standardwert) "YES" = Spoolliste anfordern. Diese wird dabei als Text-Datei in das Verzeichnis gespeichert, welches Sie in der INI-Datei des SAP-Agenten mit dem Parameter Download_dir= (Sektion [GLOBAL])festlegen. Der Name dieser Datei setzt sich folgendermaßen zusammen: *<SAP-Job-Count>_<Step-Nummer>_<Spool-Nummer>*.txt Zusätzlich wird diese Datei beim AE-Job als Job-Output registriert. "NO" = Spooliste wird nicht angefordert Wird das Sprachmittel in einem SAP-Job mehrmals verwendet und alle einzelnen Job-Steps zusammengefasst (durch die Option "Jobs-Steps kombinieren" im Verbindungs-Objekt), der Parameter GET_SPOOL="YES" aber nur für eine R3_ACTIVE_REPORT - Zeile gesetzt, so wird die Spoolliste trotzdem für alle Child-Jobs angefordert. Spool-Einträge der Children werden nur dann angefordert, wenn der Parameter REPLICATE= entweder auf YES oder ALL gesetzt ist.

Anmerkungen

Das Script-Sprachmittel ermöglicht es, einen ABAP-Step zu modifizieren. Nachdem der SAP-Job selektiert wurde, können für einen ABAP-Step der Name des Reports, der Name der Variante und verschiedene Archivierungs- und Druckparameter neu gesetzt werden.

Die Parameter entsprechen den Feldern der SAP-Dictionary Struktur PRI_PARAMS und ARC_PARAMS. Verwenden Sie bitte Dictionary oder BAPI-Browser, um detailliertere Informationen zu den einzelnen Feldern zu erhalten.

Anstelle einer Variante können Sie die Selektionskriterien auch mit dem Script-Sprachmittel R3_SET_SELECT_OPTION übergeben.

Beispiel

```
R3_ACTIVATE_REPORT REPORT='ZSUSER00',VARIANT='ALL_
USERS',COVERPAGE='YES',DESTINATION='LT77',IMMEDIATELY='YES'
```

Siehe auch:

Allgemeines zur SAP-JCL
Beispielsammlung
Aufruf eines MBeans

R3_ACTIVATE_SESSIONS

Abspielen von Batch-Input-Mappen.

Transaktion: SM35

Schnittstelle: AE

Syntax

R3_ACTIVATE_SESSION[**S**]

NAME=...
[**,STATUS=**...]
[**,NOFOUND=**...]
[**,ERROR=**...]
[**,ERRORLEVEL=**...]
[**,SELECT=**...]
[**,JOBNAME=**...]
[**,ORDER_BY=**...]

Syntaxteil	Beschreibung/Format
NAME=	Selektion von Mappen nach Namen (z.B. 'xxx*') Format: Script-Literal
STATUS=	Selektion von Mappen nach Status (z.B. alle noch zu verbuchenden Mappen) Format: Script-Literal Erlaubte Werte: "" (Standardwert) und "E" "" - Mappe ist noch zu bearbeiten. "E" - fehlerhafte Mappe.
NOFOUND=	Behandlung wenn durch die Selektion keine Mappen getroffen wurden Format: Script-Literal Erlaubte Werte: "NORMAL" (Standardwert) und "ABEND" "NORMAL" - Das Script wird fortgesetzt, der AE Job endet normal. "ABEND" - Das Script wird nicht fortgesetzt, der AE Job endet abnormal.

ERROR=	Behandlung, wenn eine Mappe aus der selektieren Menge abnormal endet Format: Script-Literal Erlaubte Werte: "ABEND" (Standardwert) und "IGNORE" "ABEND" - Das Script wird nicht fortgesetzt, der AE Job endet abnormal "IGNORE" - Das Script wird fortgesetzt, der AE Job endet normal. Siehe:ERROR/ERRORLEVEL bei R3_ACTIVATE_SESSIONS
ERRORLEVEL=	Gibt die Anzahl an fehlerhaften Transaktionen in Prozent (%) an Format: Script-Literal Standardwert: "101" Wird diese Anzahl je Mappe überschritten, so endet der AE Job abnormal.
SELECT=	Einmalig oder permanent selektieren Format: Script-Literal Erlaubte Werte "ONCE" (Standardwert) und "EVERY" "ONCE" - Es wird die Treffermenge (= Anzahl freizugebender Mappen) einmalig ermittelt. "EVERY" - Nach jeder Freigabe einer Mappe wird die Treffermenge erneut ermittelt. Diese Technik erlaubt das Parallelisieren freigegebener Mappen.
JOBN[AME]=	Jede Mappe wird mit einem SAP Job abgespielt Format: Script-Literal Erlaubte Werte: "ATTRIBUTE" (Standardwert) und "SESSION" "ATTRIBUTE" = Inhalt des Textfeldes "Jobname" aus der Registerkarte **Host-Attribute**. Ist dieses leer, wird ein Standard-Jobname von AE gebildet. "SESSION" = Name der Mappe, die abgespielt wird.
ORDER[_BY]=	Kriterium nach der die Selektion der Mappen sortiert wird. Es können alle Feldnamen der SAP Tabelle APQI angegeben werden. Beispiel: ORDER_BY=GROUPID Der Parameter wird ab SAP-Version 4.6 unterstützt.

R3_ACTIVATE_SESSION[**S**]

[**QID=**...]
[**,NOFOUND=**...]
[**,ERROR=**...]
[**,ERRORLEVEL=**...]
[**,JOBNAME=**...]

Syntaxteil	Beschreibung/Format
QID=	Queue-ID, mit der eine Batch-Input-Mappe eindeutig identifiziert wird Format: Script-Literal Parameter wird verwendet, um eine einzelne Batch-Input-Mappe zu starten.

NOFOUND=	Behandlung wenn durch die Selektion keine Mappen getroffen wurden Format: Script-Literal Erlaubte Werte: "NORMAL" (Standardwert) und "ABEND" "NORMAL" - Das Script wird fortgesetzt, der AE Job endet normal. "ABEND" - Das Script wird nicht fortgesetzt, der AE Job endet abnormal.
ERROR=	Behandlung, wenn eine Mappe aus der selektieren Menge abnormal endet Format: Script-Literal Erlaubte Werte: "ABEND" (Standardwert) und "IGNORE" "ABEND" - Das Script wird nicht fortgesetzt, der AE Job endet abnormal "IGNORE" - Das Script wird fortgesetzt, der AE Job endet normal. Siehe:ERROR/ERRORLEVEL bei R3_ACTIVATE_SESSIONS
ERRORLEVEL=	Gibt die Anzahl an fehlerhaften Transaktionen in Prozent (%) an Format: Script-Literal Standardwert: "101" Wird diese Anzahl je Mappe überschritten, so endet der AE Job abnormal.
JOBN[AME]=	Jede Mappe wird mit einem SAP Job abgespielt Format: Script-Literal Erlaubte Werte: "ATTRIBUTE" (Standardwert) und "SESSION" "ATTRIBUTE" = Inhalt des Textfeldes "Jobname" aus der Registerkarte **Host-Attribute**. Ist dieses leer, wird ein Standard-Jobname von AE gebildet. "SESSION" = Name der Mappe, die abgespielt wird.

Beschreibung

Die zu startenden Batch-Input-Mappen können über eine Selektion ermittelt werden. Mit dem Parameter QID= kann auch eine einzelne Batch-Input-Mappe gestartet werden. Aus dem Selektionsergebnis von R3_GET_SESSIONS, welches im Aktivierungsreport oder in einer Datei gespeichert wurde, lässt sich die Queue-ID einer einzelnen Batch-Input-Mappe herausfinden.

Mit dem Parameter ORDER_BY kann die Reihenfolge der Abarbeitung der Batch-Input-Mappen gesteuert werden, da die Selektion nach dem angegebenen Kriterium sortiert wird.

Beispiele

Alle Mappen mit Namen 'FI*' abspielen, welche noch zu bearbeiten sind. Den Job abnormal beenden, wenn fehlerhafte Transaktionen enthalten sind.

```
R3_ACTIVATE_SESSIONS
NAME='FI*',STATUS=,NOFOUND='NORMAL',ERROR='ABEND',ERRORLEVEL=0
```

Im zweiten Beispiel wird eine einzelne Batch-Input-Mappe gestartet, deren Queue-ID als Parameter angegeben wurde.

```
R3_ACTIVATE_SESSION QID='20020318171302022315'
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_CALL_TRANSACTION

Ruft eine SAP-Transaktion auf.

Transaktion: -

Schnittstelle: UC4

Syntax

R3_CALL_TRANS[ACTION]

CODE=...
[**,UPDATE=**...]
[**,RACOMMIT=**...]
[**,NOBINPT=**...]

Syntaxteil	Beschreibung/Format
CODE=	Transaktionscode, 20 Stellen Format: Script-Literal
UPDATE=	Verbuchermodus für diese Transaktion Format: Script-Literal Erlaubte Werte: "A", "L" und "S" (Standardwert) "A" - Asynchron. "L" - Lokal. "S" - Synchron.
RACOMMIT=	Ende der Transaktion bei Commit Work Format: Script-Literal Erlaubte Werte: "" (Standardwert) und "X" "" - Transaktion läuft bis zum normalen Transaktionsende. "X" - Transaktion wird im SAP beendet, sobald das ABAP-Sprachmittel COMMIT WORK durchlaufen wurde.
NOBINPT=	Verarbeitung im Batch-Input Modus Format: Script-Literal Erlaubte Werte: "" (Standardwert) und "X" "" - Transaktion wird im BDC-Modus verarbeitet. "X" - Transaktion wird im SAP so verarbeitet, als ob sie von einem Dialog-Benutzer gestartet worden ist.

Anmerkungen

Das Script-Sprachmittel ruft eine SAP-Transaktion auf. Die für die Transaktion erforderlichen Daten wurden zuvor mit R3_SET_BDCDATA definiert.

Aus Sicherheitsgründen wird die Transaktion nicht mit den Rechten des RFC-Benutzers (CPIC) durchgeführt, mit dem sich die Automation Engine standardmäßig an SAP anmeldet und Hintergrundjobs einplant. Der Agent meldet sich für den Aufruf der Transaktion vom SAP-System ab und unter dem Benutzer wieder an, der im Login-Objekt für diesen AE Job definiert wurde. Wir empfehlen, in SAP einen speziellen Benutzer anzulegen, der für den Aufruf der Transaktionen aus der Automation Engine heraus verwendet wird. Dieser Benutzer muss Transaktionsberechtigungen haben.

Es ist auch sinnvoll, die Script-Sprachmittel R3_SET_BDCDATA und R3_CALL_TRANSACTION in eigenen AE Jobs einzusetzen. Dadurch wird vermieden, dass der für R3_CALL_TRANSACTION definierte Benutzer, auch andere Script-Sprachmittel der SAP-JCL, beispielsweise R3_ ACTIVATE_REPORT, beeinflusst.

Beispiel

Im Beispiel werden die BDC-Daten für die Transaktion "SA38" definiert. Anschließend wird die Transaktion "SA38" aufgerufen, um ein Update der Daten durchzuführen.

```
R3_SET_BDCDATA PROGRAM="SAPMS38M", DYNPRO="0101", DYBEGIN="X"
R3_SET_BDCDATA FNAM="BDC_OKCODE", FVAL="=SSET"
R3_SET_BDCDATA FNAM="RS38M-PROGRAMM", FVAL="RSEINB00"
R3_SET_BDCDATA PROGRAM="SAPLSVAR", DYNPRO="0302", DYBEGIN="X"
R3_SET_BDCDATA FNAM="BDC_OKCODE", FVAL="=CHNG"
R3_SET_BDCDATA FNAM="RSVAR-VARIANT", FVAL="UM-V1"
R3_SET_BDCDATA FNAM="RSVAR-FLAG1", FVAL="X"
R3_SET_BDCDATA PROGRAM="RSEINB00", DYNPRO="1000", DYBEGIN="X"
R3_SET_BDCDATA FNAM="BDC_OKCODE", FVAL="=SAVE"
R3_SET_BDCDATA FNAM="P_FILE", FVAL="test.txt"
R3_SET_BDCDATA PROGRAM="RSEINB00", DYNPRO="1000", DYBEGIN="X"
R3_SET_BDCDATA FNAM="BDC_OKCODE", FVAL="=VBAC"
R3_SET_BDCDATA PROGRAM="SAPLSVAR", DYNPRO="0302", DYBEGIN="X"
R3_SET_BDCDATA FNAM="BDC_OKCODE", FVAL="/EBACK"
R3_SET_BDCDATA PROGRAM="SAPMS38M", DYNPRO="0101", DYBEGIN="X"
R3_SET_BDCDATA FNAM="BDC_OKCODE", FVAL="=BACK"
R3_CALL_TRANSACTION CODE="SA38", UPDATE="S"
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_COPY_VARIANT

Kopiert die Variante eines Reports.

Transaktion: SA38

Schnittstelle: AE und Standard

Syntax

R3_COPY_VARIANT

REP[ORT]=...
,S[OURCE]=...
,T[ARGET]=...
[**,DELAY=**...]
[,**MODE=**...]
[,**OVERWRITE=**...]
[,**PROTECTED=**...]

Syntaxteil	Beschreibung/Format	AE-Schnittstelle erforderlich
REP[ORT]=	Name des Reports, dessen Variante kopiert werden soll Format des Wertes: Script-Literal	
S[OURCE]=	Name der Quelle-Variante Format des Wertes: Script-Literal	
T[ARGET]=	Name der Ziel-Variante Format des Wertes: Script-Literal	
DELAY=	Zeitspanne in Sekunden, die der Agent nach dem Kopieren einer Variante wartet Format des Wertes: Script-Literal Standardwert: "0" Wird verwendet, um bei einem SAP-System mit mehreren Applikationsservern mögliche Synchronisationsprobleme (soeben kopierte Variante wird nicht gefunden) zu umgehen.	
MODE=	Verarbeitungsmodus Format des Wertes: Script-Literal Erlaubte Werte: "" (Standardwert) und "C" "" - Kopieren der Variante durch Duplizieren der Tabelleneinträge. "C" - Neuanlegen der Variante mit Referenz mittels SAP-internen Schnittstellen. Soll eine Systemvariante kopiert werden, müssen Sie den Parameter MODE=C angeben. Eine Kopie derselben wird nicht automatisch in einen Transportauftrag eingetragen. Der Parameter wird bei der Standard-Schnittstelle nicht benötigt.	✓
OVERWRITE=	Behandlung, wenn die Variante bereits vorhanden ist. Format des Wertes: Script-Literal Erlaubte Wert: "YES" und "NO" (Standardwert) "YES" - Die Zielvariante wird überschrieben. "NO" - Es erfolgt keine Überschreibung.	

Syntaxteil	Beschreibung/Format	AE-Schnittstelle erforderlich
PROTECTED=	Handhabung der Variante während des Kopiervorganges. Erlaubte Werte: "YES" und "NO" (default value) "YES" - Die Variante ist geschützt und kann nicht geändert werden. "NO" - Die Variante ist nicht geschützt.	

⚠ Beachten Sie beim Einsatz der Schnittstelle XBP3.0, dass die Optionen "Variante schützen" und "Nur für die Hintergrundverarbeitung" beim Kopiervorgang nicht übernommen werden und die kopierte Variante für alle Benutzer (Batch und Dialog) sichtbar wird!

Beispiel

Die Variante 'DAILY' des Reports 'RSPO0041' als Variante 'DAILYCOP' kopieren.

```
R3_COPY_VARIANT REPORT='RSPO0041',SOURCE='DAILY',TARGET='DAILYCOP'
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_CREATE_OUTPUT_REQUEST

Erzeugt zu einem bestehenden Spoolauftrag einen neuen Ausgabeauftrag.

Transaktion: SP01

Schnittstelle: UC4

Syntax

R3_CREATE_OUTPUT_REQ[UEST]

SPOOLNR=...
[**,DESTINATION=**...]
[**,RECEIVER=**...]
[**,DEPARTMENT=**...]
[**,COPIES=**...]
[**,SPOOL_PRIORITY=**...]
[**,TITLE=**...]
[**,PAGE_FROM=**...]
[**,PAGE_TO=**...]
[**,ERROR=**...]

Syntaxteil	Beschreibung/Format
SPOOLNR=	Nummer des Spoolauftrags Format des Wertes: Script-Literal

DEST[INATION]=	Ausgabegerät Format des Wertes: Script-Literal
RECEIVER=	Empfänger Format des Wertes: Script-Literal
DEPART[MENT]=	Abteilung Format des Wertes: Script-Literal
COPIES=	Anzahl der Kopien Format des Wertes: Zahl Standardwert: "0"
SPOOL_PRI[ORITY]=	Spool-Priorität Format des Wertes: Script-Literal oder Zahl Standardwert: "0"
TITLE=	Spool-Titel Format des Wertes: Script-Literal
PAGE_FROM=	Von Seite Format des Wertes: Zahl Standardwert: "0"
PAGE_TO=	Bis Seite Format des Wertes: Zahl Standardwert: "0"
ERROR=	Behandlung, wenn ein Fehler auftritt. Format des Wertes: Script-Literal Erlaubte Werte: "NORMAL" und "ABEND" (Standardwert) "NORMAL" - Der AE-Job läuft weiter. "ABEND" - der AE-Job endet abnormal

Beispiel

```
R3_CREATE_OUTPUT_REQUEST
SPOOLNR='1234',DESTINATION='PRNT',RECEIVER='MEIER',DEPARTMENT='UC4',CO
PIES=1,TITLE='Auswertung Dezember'
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_CREATE_VARIANT

Legt eine neue Variante an.

Transaktion: SA38

Schnittstelle: Standard

Syntax

R3_CREATE_VAR[IANT]

REP[ORT]=...
,VAR[IANT]=...
[**,TEXT=**...]
[**,PROTECTED=**...]

Syntaxteil	Beschreibung/Format
REP[ORT]=	Name des Reports. Format: Script-Literal
VAR[IANT]=	Name für die Variante. Format: Script-Literal
TEXT=	Variantenkurztext. Format: Script-Literal Der Text wird in der Spracheinstellung des Agenten, die der Administrator in der INI-Datei konfiguriert, oder jener des Jobs angelegt. Letztere übersteuert die Option in der INI-Datei. Wenn Sie diesen Parameter nicht angeben, wird der Name der Variante als Kurztext verwendet.
PROTECTED=	Schutz vor Änderungen. Format: Script-Literal Erlaubte Werte: "YES" und "NO" (Standardwert) "YES" - Nur der CPIC-User kann die Variante nach der Erstellung ändern. "NO" - Die Variante lässt sich uneingeschränkt modifizieren.

Anmerkungen

Die Automation Engine setzt beim Anlegen automatisch das Attribut "Variante für Hintergrundverarbeitung". Die restlichen Eigenschaften der Variante werden so gesetzt, als würde Sie jene mit dem SAP GUI erstellen.

Die Funktion R3_CREATE_VARIANT legt die Variante immer ohne Inhalt an. Verwenden Sie R3_SET_SELECT_OPTION, um deren Inhalt zu befüllen. Möchten Sie einen Report mit einer neuen Variante ausführen, ist folgende Reihung beim Aufrufen der Funktionen zu berücksichtigen:

1. R3_SET_SELECT_OPTION - Selektionskriterien definieren
2. R3_CREATE_VARIANT - Variante anlegen
3. R3_ACTIVATE_REPORT - Report mit dieser Variante ausführen

Falls die Variante bereits existiert und neu angelegt werden soll, so können Sie diese zuvor mit der Funktion R3_DELETE_VARIANT löschen.

Der Job wird abgebrochen, wenn beim Anlegen der Variante ein Fehler auftritt.

Der Mandant der Variante ist automatisch jener Mandant in dem sich der SAP-Agent aufgrund des Login-Objektes angemeldet hat. SAP Systemvarianten (CUS& und SAP&) können ebenfalls angelegt werden. Dabei wird der Mandant der Variante automatisch immer auf 000 gesetzt.

Voraussetzung für den Einsatz von R3_CREATE_VARIANT ist XBP 2.0 oder höher (SAP Release 4.6+).

Beispiel

Das Beispiel definiert den Wert "17" für den Parameter "Mindestalter". Das Selektionskriterium wird anschließend für das Anlegen einer Variante verwendet.

```
R3_SET_SELECT_OPTION SELNAME='MIN_ALT',KIND='P',LOW='17',SIGN='I'
R3_CREATE_VARIANT REP=REPORT01,VAR=NEW,TEXT='Neue Variante'
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_DEACTIVATE_CM_PROFILE

Deaktiviert ein Profil im SAP Kriterienmanager.

Transaktion: SM62

Schnittstelle: Standard

Syntax

R3_DEACTIVATE_CM_PROFILE

TYPE=...
[,**ERROR**=...]

Syntaxteil	Beschreibung/Format
TYPE=	Typ des Profils Format: Script-Literal Erlaubte Wert: "EVHIRO", "EVTHIS" und "INTERC" "EVHIRO" - Event History Reorg "EVTHIS" - Event History "INTERC" - Job Interception
ERROR=	Behandlung, wenn das Profil nicht deaktiviert werden kann Format des Wertes: Script-Literal Erlaubte Werte: "ABEND" (Standardwert) und "IGNORE" "ABEND" - Das Script wird nicht fortgesetzt, der AE Job endet abnormal. "IGNORE" - Das Script wird fortgesetzt, der AE Job endet normal.

Anmerkungen

Pro Profiltyp kann nur ein Profil aktiviert sein. Deshalb brauchen Sie nicht die ID angeben, sondern nur den Typ.

Beispiel

Das Beispiel deaktiviert das aktive Profil für die Event History.

```
R3_DEACTIVATE_CM_PROFILE TYPE="EVTHIS"
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_DELETE_NODE

Löscht einen Knoten in der SAP-Monitor-Architektur.

Transaktion: RZ20

Schnittstelle: XMW

Syntax

R3_DELETE_NODE

NODE=...

Syntaxteil	Beschreibung/Format
NODE=	Name des Knotens Format des Wertes: Script-Literal

Anmerkungen

Das Script-Sprachmittel löscht einen existierenden Knoten (Attribut-, Objekt-, Summen- oder Kontextknoten). Sollte der Knoten noch offene Alarme besitzen, werden diese vor dem Löschen geschlossen.

Format des Parameters NODE=

Dieser Parameter beschreibt einen kompletten Pfad. Die einzelnen Bestandteile sind durch Schrägstriche "/" getrennt.

Der Pfad beginnt immer mit einem Kontextknoten. Danach können Summenknoten, ein Objektknoten und ein Attributknoten folgen.

Kontext- , Objekt- und Attributknoten dürfen nur einmal im Pfad vorkommen. Summenknoten hingegen mehrmals.

Beispiel 1:

"UC4/TestNode/PerfAttributUC4"

- "UC4" - Kontextname
- "TestNode" - Objektknoten
- "PerfAttributUC4" - Attributknoten

Beispiel 2:

"UC4/Summary1/Summary2/TestNode/PerfAttributUC4"

- "UC4" - Kontextname
- "Summary1" - Summenknoten
- "Summary2" - Summenknoten
- "TestNode" - Objektknoten
- "PerfAttributUC4" - Attributknoten

Beispiel

Nach dem Ausführen des folgenden Beispiels existiert noch der Knoten "UC4/TEST". Der Subknoten "PerfAttributUC4" wurde gelöscht.

```
R3_DELETE_NODE NODE='UC4/TEST/PerfAttributUC4'
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_DELETE_VARIANT

Löscht die Variante eines Reports.

Transaktion: SA38

Schnittstelle: AE und Standard

Syntax

R3_DELETE_VARIANT

REP[ORT]=...
,VAR[IANT]=...

Syntaxteil	Beschreibung/Format
REP[ORT]=	Name des Reports, dessen Variante gelöscht werden soll Format: Script-Literal
VAR[IANT]=	Name der Variante, welche gelöscht werden soll Format: Script-Literal

Beispiel

Die Variante 'DAILYCOP' des Reports 'RSPO0041' löschen.

```
R3_DELETE_VARIANT REPORT='RSPO0041',VAR='DAILYCOP'
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_GET_APPLICATION_RC

Prüft den Anwendungsreturncode eines oder mehrerer Job-Steps.

Transaktion: -

Schnittstelle: Standard (XBP 3.0)

Syntax

R3_GET_APPLICATION_RC

NAME=...
,JOBCOUNT=...
[**,STEP=**...]
[**,MAX_APPL_RC=**...]

Syntaxteil	Beschreibung/Format
NAME=	Name des SAP-Jobs Format des Wertes: Script-Literal Standardmäßig wird der SAP-Job verwendet, den Sie zuvor im Script in einem der folgenden Script-Sprachmittel angegeben haben: • R3_ACTIVATE_INTERCEPTED_JOBS • R3_ACTIVATE_JOBS • R3_ACTIVATE_REPORT • R3_ACTIVATE_SESSION
JOBCOUNT=	Nummer des SAP-Jobs. Format: Script-Literal In Verbindung mit NAME= wird der SAP-Job eindeutig identifiziert.
STEP=	Nummer des Steps im SAP-Job Format des Wertes: Zahl Standardwert: "1" Wenn Sie "0" angeben, werden die Anwendungsreturncodes von allen Steps des angegebenen SAP-Jobs geprüft.
MAX_APPL_RC=	Maximal erlaubter Anwendungsreturncode Format des Wertes: Zahl Standardwert: "0" Der AE-Job bricht ab, wenn der Anwendungsreturncode eines Job-Steps den hier angegebenen Wert überschreitet.

Anmerkungen

Das Script-Sprachmittel prüft die Anwendungsreturncodes von Job-Steps und vergleicht sie mit dem maximal erlaubten Wert. Wird dieser überschritten, so bricht der AE-Job ab. Wenn dies nicht der Fall ist, wird die Jobdurchführung fortgesetzt.

Der Anwendungsreturncode ist auch im Report ersichtlich und kann ausgelesen werden.

Beispiel

Im ersten Beispiel wird der Anwendungsreturncode des Steps "2" geprüft.

```
R3_GET_APPLICATION_RC NAME="MYJOB",JOBCOUNT=13541601,STEP=2,MAX_APPL_
RC=10
```

Das zweite Beispiel zeigt wie alle Anwendungsreturncodes des Jobs "MYJOB" geprüft werden.

```
R3_GET_APPLICATION_RC NAME="MYJOB",JOBCOUNT=13541601,STEP=0,MAX_APPL_
RC=10
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_GET_APPLICATIONLOG

Holt Meldungen aus dem Anwendungs-Log (Transaktion SLG1) und gibt sie in Report oder Datei aus.

Transaktion: SLG1

Schnittstelle: UC4

Syntax

R3_GET_APPLICATIONLOG

[**OBJECT=**...]
[**,SUBOBJECT=**...]
[**,FROM_DATE=**...]
[**,FROM_TIME=**...]
[**,TO_DATE=**...]
[**,TO_TIME=**...]
[**,USER=**...]
[**,TAC=**...]
[**,PROGRAM=**...]
[**,MESSAGE_CLASS=**...]
[**,FILE=**...]
[**,EXTNUMBER=**...]
[**,LOG_CREATION=**...]

Syntaxteil	Beschreibung/Format
OBJ[ECT]=	Name eines Objektes (Applikationskürzel) Format: Script-Literal Standardwert: "*" Die Verwendung von "*" als Wildcard für beliebige Zeichen ist möglich.

SUB[OBJECT]=	Name eines Unterobjektes zum Objekt Format: Script-Literal Standardwert: "*" Die Verwendung von "*" als Wildcard für beliebige Zeichen ist möglich.
FROM_D[ATE]=	Start-Datum der Selektion von Meldungen im Format "JJJJMMTT" Format: Script-Literal Standardwert: "20010101"
FROM_T[IME]=	Start-Zeit der Selektion von Meldungen im Format "HHMMSS" Format: Script-Literal Standardwert: "000000"
TO_D[ATE]=	Ende-Datum der Selektion von Meldungen im Format "JJJJMMTT" Format: Script-Literal Standardwert: Aktuelles Datum
TO_T[IME]=	Ende-Zeit der Selektion von Meldungen im Format "HHMMSS" Format: Script-Literal Standardwert: "235959"
USER=	Name des Benutzers, der ein protokolliertes Ereignis ausgelöst hat Format: Script-Literal Standardwert: "*" Die Verwendung von "*" als Wildcard für beliebige Zeichen ist möglich.
TAC=	Name der Transaktion, mit der das protokollierte Ereignis ausgelöst wurde Format: Script-Literal Standardwert: "*" Die Verwendung von "*" als Wildcard für beliebige Zeichen ist möglich.
PROGRAM=	Name des Programms, welches das protokollierte Ereignis ausgelöst hat Format: Script-Literal Standardwert: "*" Die Verwendung von "*" als Wildcard für beliebige Zeichen ist möglich.
MESSAGE_CL[ASS]=	Kennzeichen für die Zugehörigkeit einer Meldung zu einer Problemklasse Format: Script-Literal Standardwert: "4" Die Kennzeichen von Meldungen sind Werte von "1" (sehr wichtig) bis "4" (Zusatzinformation).
FILE=	Name einer Datei, in welcher das Ergebnis (gefundene Meldungen) gespeichert werden soll Format: Script-Literal Das Selektionsergebnis wird nicht im Report ausgegeben, wenn dieser Parameter verwendet wird.

EXTNUMBER=	Externe, vom Anwendungsprogramm vergebene Identifikation des Anwendungs-Logs Format: Script-Literal
LOG_C[REATION]=	Betriebsmodus, in dem das Anwendungs-Log erzeugt wurde Format: Script-Literal Erlaubte Werte: "" (Standardwert), "B", "D" und "I" "" - beliebig "B" - Batch "D" - Dialog "I" - Batch-Input
ENCODING=	Encoding für die erzeugte Ausgabedatei (Parameter FILE=). Beispiel: UTF-8 Standardwert: ISO-8859-1 Wird ein nicht unterstütztes bzw. ungültiges Encoding angegeben, so führt dies zum Abbruch des Jobs mit einer entsprechenden Fehlermeldung. In den SAP-Forms ist für dieses Feld eine Eingabehilfe verfügbar, welche alle unterstützten Encodings auflistet.

Anmerkungen

Mit dem Script-Sprachmittel können Meldungen nach bestimmten Kriterien aus dem Anwendungs-Log selektiert werden. Die gefundenen Meldungen werden standardmäßig im AE-Report ausgegeben. Alternativ können sie auch in eine Datei geschrieben werden, die mit dem Parameter FILE= festgelegt wurde.

Das Script-Sprachmittel ermöglicht beispielsweise den Zugriff auf das Anwendungs-Log des SAP Banking - Accounts Managements. Meldungen von Prozessen und Prozessnetzwerken, die mit BCA_ACTIVATE_PROCESS gestartet wurden, sind somit in der AE auswertbar.

Die Parameter des Script-Sprachmittels sollten so verwendet werden, dass stets auf eine überschaubare Anzahl von Meldungen aus dem Anwendungs-Log zugegriffen wird. Durch genaue Selektionskriterien kann ein unerwünscht großes Datenvolumen vermieden werden.

Erzeugte Dateien werden standardmässig am Rechner, auf dem der Agent installiert ist, gepeichert (z.B.: R3_GET_JOB_SPOOL; FILE=).

Siehe auch:

Allgemeines zur SAP-JCL

R3_GET_EVENT

Wartet auf ein in SAP ausgelöstes Ereignis.

Transaktion: SM36 (Startbedingungen)

Schnittstelle: Standard

Syntax

R3_GET_EVENT

ID=...
[,**PARAM**=...]
[,**TIMEOUT**=...]

Syntaxteil	Beschreibung/Format
ID=	Name des Ereignisses. Format: Script-Literal Erlaubt ist eine Zeichenkette mit maximal 32 Zeichen.
PARAM=	Parameter für das Ereignis. Format: Script-Literal Erlaubt ist eine Zeichenkette mit maximal 64 Zeichen.
TIMEOUT=	Zeit in Sekunden, wie lange auf ein in SAP ausgelöstes Ereignis gewartet werden soll. Format: Zahl Standardwert: "0"

Anmerkungen

Mit dem Script-Sprachmittel kann auf ein Ereignis gewartet werden, welches in SAP ausgelöst wurde. Dabei kann es sich um ein Systemereignis - ein von SAP vordefiniertes Ereignis - oder um ein Benutzerereignis handeln. Als Ereignis-Parameter kann eine Zeichenkette mit beliebigem Inhalt angegeben werden. Wird die Zeit, wie lange auf das Ereignis gewartet werden soll, mit null Sekunden definiert, wartet das Script-Sprachmittel endlos auf das Ereignis.

Nachdem das Ereignis auftrat, wird das Script weiter prozessiert.

⚠ Beachten Sie, dass das Sprachmittel nur reagieren kann, wenn sich das angegebene Ereignis auch in der SAP Event History befindet. Passen Sie dazu eventuell die Ereignis-Kriterienprofile in SAP entsprechend an.

Beispiel

Im Beispiel soll auf das Ereignis "TEST" mit dem Parameter "Myparam" maximal 10 Sekunden gewartet werden.

```
R3_GET_EVENT ID="TEST",PARAM="Myparam",TIMEOUT=10
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_GET_INTERCEPTION

Liest die Filtertabelle für Intercepted Jobs aus und speichert diese im Aktivierungsprotokoll oder in einer Datei.

Transaktion: -

Schnittstelle: Standard (XBP 2.0)

Syntax

R3_GET_INTERC[EPTION]

[**FILE=**]

Syntaxteil	Beschreibung/Format
FILE=	Name einer Datei, in welcher die Intercepted Jobs gespeichert werden sollen. Format des Wertes: Script-Literal Das Ergebnis wird nicht im Aktivierungsprotokoll ausgegeben, wenn dieser Parameter verwendet wird.
ENCODING=	Encoding für die erzeugte Ausgabedatei (Parameter FILE=). Beispiel: UTF-8 Standardwert: ISO-8859-1 Wird ein nicht unterstütztes bzw. ungültiges Encoding angegeben, so führt dies zum Abbruch des Jobs mit einer entsprechenden Fehlermeldung. In den SAP-Forms ist für dieses Feld eine Eingabehilfe verfügbar, welche alle unterstützten Encodings auflistet.

Anmerkungen

Das Script-Sprachmittel liest den Inhalt der Tabelle aus dem SAP-System, in welcher die Bedingungen für Batch-Jobs definiert wurden. Dabei wird der SAP-Mandant verwendet, der im Login-Objekt, auf welches der AE Job referenziert, angegeben ist.

Der Tabelleninhalt wird in das Aktivierungsprotokoll oder in eine angegebene Datei geschrieben. Die Datei hat einen strukturierten Aufbau. Die erste Spalte (4 Zeichen) enthält den SAP-Mandant, die zweite Spalte (33 Zeichen) den Jobnamen und die dritte Spalte den Namen des Benutzers (12 Zeichen).

Die erzeugten Dateien werden standardmässig am Rechner, auf dem der Agent installiert ist, gepeichert (z.B.: R3_GET_JOB_SPOOL; FILE=).

Beachten Sie, dass das Script-Sprachmittel nur beim Einsatz von XBP 2.0 unterstützt wird!

Beispiel

Es werden die definierten Intercepted Jobs abgefragt und in eine Datei geschrieben.

```
R3_GET_INTERCEPTION FILE='C:\TEMP\IC.TXT'
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_GET_JOB_SPOOL

Liest die Spool-Liste eines Steps vom Typ "ABAP-Programm".

Transaktion: SM37

Schnittstelle: AE und Standard

Syntax

R3_GET_JOB_SPOOL

FILE=...
[**,NAME=**...]
[**,JOBCOUNT=**...]
[**,STEP=**...]
[**,NOFOUND=**...]
[**,MAXLINES=**...]
[**,SPOOLNR**=...]
[**,FORMAT**=...]
[**,PAGES**=...]
[**,FILTER**=...]
[**,RAW**=...]

Syntaxteil	Beschreibung/Format
FILE=	Name einer Datei, in welche das Ergebnis (Spool-Liste des Steps) geschrieben werden soll Format des Wertes: Script-Literal Zusätzlich wird diese Datei beim AE-Job als Job-Output registriert.
NAME=	Jobname des SAP-Jobs, welcher den Step enthält Format des Wertes: Script-Literal Wird der Jobname nicht definiert, so wird automatisch jener Job verwendet, der zuletzt über den AE-Job durchgeführt wurde.
JOBCOUNT=	Jobcount des SAP-Jobs, welcher den Step enthält Format des Wertes: Script-Literal Wird der Jobcount nicht angegeben, so wird automatisch der Wert des Jobs verwendet, der zuletzt über den AE-Job gelaufen ist. Beachten Sie bei Verwendung der Parameter NAME und JOBCOUNT, dass der Spool nicht ermittelt werden kann, wenn der betreffende Job im SAP-System gelöscht wurde (Einstellung "Job nach Durchführung im CCMS löschen" im AE-Job).
STEP=	Nummer des Steps im SAP-Job Format des Wertes: Zahl Standardwert: "1"

Syntaxteil	Beschreibung/Format
NOFOUND=	Behandlung, wenn die Spool-Liste nicht gefunden wurde Format des Wertes: Script-Literal Erlaubte Werte: "NORMAL" (Standardwert) und "ABEND" "NORMAL" - Der AE-Job läuft weiter. "ABEND" - der AE-Job endet abnormal.
MAXLINES=	Maximale Anzahl von Zeilen, die in die Datei geschrieben werden Format des Wertes: Zahl Standardwert: "9999" Diese Einstellung wird bei Verwendung von FORMAT = "PDF" oder "BIN" nicht berücksichtigt.
SPOOLNR=	Spool-Request-Nummer Format des Wertes: Script-Literal Erfolgt weder die Angabe von NAME, JOBCOUNT oder SPOOLNR, so wird automatisch der SAP-Job verwendet, der zuletzt über den AE-Job durchgeführt wurde.
FORMAT=	Ausgabeformat Format des Wertes: Script-Literal Erlaubte Werte: "TXT" (Standardwert), "RAW" (statt Parameter RAW), "PDF", "BIN" und "HTM" Bei den Einstellungen "BIN" und "PDF" wird die Spool-Liste unkonvertiert im Binärformat übertragen.
PAGES=	Seiten Format des Wertes: Script-Literal Syntax des Parameters: **PAGES=***Seite*[;*Seite*[;...]] Die Angabe zu den einzelnen Seiten bzw. den Seitenbereichen sind mit ";" zu trennen. Beispiele: • Seite 7: `PAGES="7"` • Seiten 5 bis 7: `PAGES="5,7"` • Seite 2 und die Seiten 5 bis 7: `PAGES="2;5,7"` Anstatt der letzen Seite kann das Symbol $ verwendet werden: • Seiten 1 bis 3 und die letzte Seite: `PAGES="1,3;$"` • alle Seiten: `PAGES="1,$"` • die erste und die letzte Seite: `PAGES="1;$"` • die letzten beiden Seiten: `PAGES="$-1,$"` Die Parameter FORMAT (ausser PDF), SPOOLNR und PAGES sind auch für die Schnittstelle XBP 3.0 (ab Version 1.1) verfügbar.

Syntaxteil	Beschreibung/Format
FILTER=	Filtereinstellung Format des Wertes: Script-Literal Zeichenfolge nach der selektiert werden soll. Die Angabe von Wildcards ist nicht möglich!
RAW=	Lieferung der Spool-Liste im Rohformat Format des Wertes: Script-Literal Erlaubte Werte: "YES" und "NO" (Standardwert) "YES" - Die Spool-Liste wird im Rohformat, d.h. inklusive aller Aufbereitungszeichen geliefert. "NO" - Die Spool-Liste ist nicht im Rohformat. Es wird empfohlen, FORMAT=RAW anstelle dieses Parameters zu verwenden.
ENCODING=	Encoding für die erzeugte Ausgabedatei (Parameter FILE=). Beispiel: UTF-8 Standardwert: ISO-8859-1 Wird ein nicht unterstütztes bzw. ungültiges Encoding angegeben, so führt dies zum Abbruch des Jobs mit einer entsprechenden Fehlermeldung. In den SAP-Forms ist für dieses Feld eine Eingabehilfe verfügbar, welche alle unterstützten Encodings auflistet.

Anmerkungen

Das Sprachmittel kann mit folgenden Parameter-Kombinationen verwendet werden:

- SPOOLNR
- NAME, JOBCOUNT, STEP
- NAME, JOBCOUNT
- STEP
- keinen dieser Parameter

Der Parameter FILTER benötigt die AE-Schnittstelle und der Parameter RAW die Standard-Schnittstelle!

Die Spool-Liste kann nur übertragen werden, wenn das Kontrollkästchen "Job nach Durchführung im CCMS löschen" in der Registerkarte "Host-Attribute" nicht aktiviert ist. Diese Einschränkung trifft nicht zu, wenn entweder der Parameter SPOOLNR angegeben oder als Standard-Schnittstelle XBP 3.0 (ab Version 1.1) ohne Parameter NAME, JOBCOUNT und SPOOLNR eingesetzt wird.

Beachten Sie folgenden Hinweis bei Verwendung der Standardschnittstelle mit einer niedrigeren Version als XBP3.0 V1.1:
Es wird immer die gesamte Spool-Liste zum SAP-Agenten übertragen. Die Standardfunktion der XBP-Schnittstelle bietet keine Möglichkeit, nur auf Teile der Spool-Liste zuzugreifen. Bitte beachten Sie, dass große Spool-Listen die Performance des SAP-Agenten beeinträchtigen können. Der Parameter **MAXLINES=** wird erst berücksichtigt, wenn die gesamte Spool-Liste bereits übertragen wurde.

Beachten Sie, dass erzeugte Dateien standardmäßig am Rechner, auf dem der Agent installiert ist, gepeichert werden (z.B.: R3_GET_JOB_SPOOL; FILE=).

Beispiel

Im Beispiel wird zunächst ein Report aktiviert. Anschließend wird die Spool-Liste des Steps geholt, ohne den Jobnamen oder den Jobcount als Parameter anzugeben. Dadurch wird automatisch der Job in SAP verwendet, der zuletzt über den AE-Job gestartet wurde.

```
R3_ACTIVATE_REPORT REPORT='RSPO0041',VAR='STANDARD',COVERPAGE=YES
R3_GET_JOB_SPOOL
FILE='c:\temp\spoollist.txt',NOFOUND=ABEND,MAXLINES=20
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_GET_JOBLOG

Holt das Job-Log eines SAP-Jobs aus SAP und gibt es im Report aus.

Transaktion: SM37

Schnittstelle: AE und Standard

Syntax

R3_GET_JOBLOG

NAME=...
,JOBCOUNT=...
[**,BEG_LOGLINES=**...]
[**,END_LOGLINES=**...]

Syntaxteil	Beschreibung/Format
NAME=	Name des SAP-Jobs. Format: Script-Literal
JOBCOUNT=	Nummer des SAP-Jobs. Format: Script-Literal In Verbindung mit dem Namen wird ein SAP-Job eindeutig identifiziert.
BEG_LOGLINES=	Legt fest wie viele Zeilen vom Beginn des SAP Joblogs in den AE-Jobreport übernommen werden sollen.

END_LOGLINES=	Anzahl der Zeilen, ab Ende des SAP Joblogs, die in den AE-Jobreport geschrieben werden. Wird weder der Parameter END_LOGLINES noch BEG_LOGLINES angegeben, so wird standardmäßig das komplette Joblog verwendet. Wenn nur ein Parameter angeführt wird, so werden entweder nur Zeilen vom Beginn oder Ende des Joblogs gelesen. Wenn "0" als Wert für beide Parameter eingetragen ist, wird kein Joblog übertragen. Für beide Parameter sind nur numerische Werte erlaubt.

Anmerkungen

Das Script-Sprachmittel wird dazu verwendet, um das Job-Log eines bereits gelaufenen SAP-Jobs zu ermitteln, der nicht durch die Automation Engine direkt gestartet wurde. So ist es beispielsweise möglich, auf das Job-Log von Sub-Jobs bei IS-U oder SARA zuzugreifen. Das Job-Log kann in der Registerkarte "Report" des AE Jobs eingesehen werden, der das Script-Sprachmittel verwendet.

Es kann jeweils nur das Job-Log eines einzelnen SAP-Jobs gelesen werden. Teilqualifizierte Angaben bei den Parametern sind nicht möglich.

Beispiel

Im Beispiel wird das Job-Log des Jobs "SWWERRE" und der Jobnummer "23483501" ermittelt.

```
R3_GET_JOBLOG NAME='SWWERRE',JOBCOUNT=23483501
```

Die folgende Zeilen werden im Report ausgegeben.

```
Date        Time MsgId/Nr Message
11.12.2001  00:08:35 00 516 SJob wurde gestartet
11.12.2001  00:08:35 00 550 SStep 001 gestartet (Programm RSWWERRE,
Variante , Benutzername NI)
11.12.2001  00:08:35 WI 137 S0 Workitems wurden bearbeitet
11.12.2001  00:08:35 00 517 SJob wurde beendet
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_GET_JOBS

Selektiert SAP-Jobs und listet das Ergebnis im Aktivierungsreport oder in einer Datei auf.

Transaktion: SM37

Schnittstelle: AE und Standard

Syntax

R3_GET_JOB[S]

NAME=...
[**,JOBCOUNT=**...]

[**,GROUP=**...]
[**,USER=**...]
[**,START_D[ATE]=**...]
[**,START_T[IME]=**...]
[**,END_D[ATE]=**...]
[**,END_TIME=**...]
[**,NO_DATE=**...]
[**,WITH_PRED=**...]
[**,EVENT_ID=**...]
[**,EVENT_PARM=**...]
[**,PRELIM=**...]
[**,SCHEDUL=**...]
[**,READY=**...]
[**,RUNNING=**...]
[**,FINISHED=**...]
[**,ABORTED=**...]
[**,NOFOUND=**...]
[**,FILE=**...]

Syntaxteil	Beschreibung/Format
NAME=	Selektion eines oder mehrerer Jobs nach Namen Format: Script-Literal oder AE-name In Verbindung mit der Jobnummer wird ein einzelner SAP-Job eindeutig identifiziert. Bei der Selektion mehrerer Jobs kann das Wildcardzeichen "*" verwendet werden (z.B. "xxx*").
Parameter	Weitere Selektionskriterien in Form eines Schlüsselwortes und einer Zuweisung. Nicht angegebene Parameter erhalten Standardwerte. Formate gelten für die Zuweisung zum Schlüsselwort.
JOBCOUNT=	Nummer des SAP-Jobs Format: Script-Literal In Verbindung mit dem Namen wird ein SAP-Job eindeutig identifiziert.
GROUP=	Selektion von Jobs nach Gruppen (z.B. "xxx*") Format: Script-Literal oder AE-Name Standardwert: "*"
USER=	Selektion von Jobs nach Benutzern (z.B. "xxx*") Format: Script-Literal oder AE-Name Standardwert: "*"
START_D[ATE]=	Geplantes Start-Datum für die Ausführung des Jobs Format: Script-Literal oder AE-Name Datumsformat: JJJJMMTT Standardwert: "20010101"
START_T[IME]=	Geplante Start-Zeit für die Ausführung des Jobs Format: Script-Literal oder AE-Name Zeitformat: HHMMSS Standardwert: "000000"

END_D[ATE]=	Geplantes Ende-Datum für die Ausführung des Jobs Format: Script-Literal oder AE-Name Datumsformat: JJJJMMTT Standardwert: aktuelles Datum
END_TIME=	Geplante Ende-Zeit für die Ausführung des Jobs Format: Script-Literal oder AE-Name Zeitformat: HHMMSS Standardwert: "235959"
NO_DATE=	Jobs ohne Starttermin Format: Script-Literal oder AE-Name Erlaubte Werte: "" (Standardwert) oder "X"
WITH_PRED=	Jobs mit Starttermin "nach Vorgänger" Format: Script-Literal oder AE-Name Erlaubte Werte: "" (Standardwert) oder "X"
EVENT_ID=	Jobs, die mit einem Ereignis verkettet sind: Name des Ereignisses Format: Script-Literal oder AE-Name
EVENT_PARM=	Jobs, die mit einem Ereignis verkettet sind: Parameter des Ereignisses Format: Script-Literal oder AE-Name
PRELIM=	Jobs mit Zustand "Geplant" Format: Script-Literal oder AE-Name Erlaubte Werte: "" (Standardwert) oder "X"
SCHEDUL=	Jobs mit Zustand "Freigegeben" Format: Script-Literal oder AE-Name Erlaubte Werte: "" (Standardwert) oder "X"
READY=	Jobs mit Zustand "Bereit" Format: Script-Literal oder AE-Name Erlaubte Werte: "" (Standardwert) oder "X"
RUNNING=	Jobs mit Zustand "Aktiv" Format: Script-Literal oder AE-Name Erlaubte Werte: "" (Standardwert) oder "X"
FINISHED=	Jobs mit Zustand "Fertig" Format: Script-Literal oder AE-Name Erlaubte Werte: "" (Standardwert) oder "X"
ABORTED=	Jobs mit Zustand "Abgebrochen" Format: Script-Literal oder AE-Name Erlaubte Werte: "" (Standardwert) oder "X"
NOFOUND=	Behandlung, wenn durch die Selektion keine Jobs gefunden wurden Format: Script-Literal oder AE-Name Erlaubte Werte: "NORMAL" (Standardwert) und "ABEND" "NORMAL" - Das Script wird fortgesetzt, der AE Job endet normal. "ABEND" - Das Script wird nicht fortgesetzt, der AE Job endet abnormal.

FILE=	Name einer Datei, in welcher das Ergebnis (gefundene SAP-Jobs) gespeichert werden soll Format: Script-Literal Das Selektionsergebnis wird nicht im Aktivierungsreport ausgegeben, wenn dieser Parameter verwendet wird.
ENCODING=	Encoding für die erzeugte Ausgabedatei (Parameter FILE=). Beispiel: UTF-8 Standardwert: ISO-8859-1 Wird ein nicht unterstütztes bzw. ungültiges Encoding angegeben, so führt dies zum Abbruch des Jobs mit einer entsprechenden Fehlermeldung. In den SAP-Forms ist für dieses Feld eine Eingabehilfe verfügbar, welche alle unterstützten Encodings auflistet.

Beschreibung

Das Script-Sprachmittel ermöglicht es, Hintergrundjobs im SAP zu selektieren. Die Selektionskriterien werden als Parameter übergeben. Die Parameter entsprechen den Feldern der SAP-Dictionary Struktur BAPIXMJSEL. Verwenden Sie bitte Dictionary oder BAPI-Browser, um detailliertere Informationen zu den einzelnen Feldern zu erhalten.

Das Ergebnis der Selektion wird entweder in den Aktivierungsreport oder in eine Datei geschrieben. Für jeden gefundenen SAP-Job gibt es eine Zeile. Der Aufbau der Zeilen im Aktivierungsreport und in der Datei ist unterschiedlich.

In der Datei werden Spalten verwendet. Die erste Zeile der Datei enthält Informationen zur Breite und zum Inhalt der Spalten. Laut dieser Festlegung sind die einzelnen Zeilen unterteilt. Die ersten 33 Stellen enthalten jeweils den Namen des SAP-Jobs, weitere 9 dessen Nummer.

Im Aktivierungsreport sind die Informationen innerhalb einer Zeile durch Semikolon getrennt. Zusätzlich wird ein Zeitstempel und eine AE-Meldungsnummer am Anfang der Zeile ausgegeben.

Das Ergebnis der Selektion kann mit Script-Funktionen analysiert werden. PREP_PROCESS_REPORT kann in der Registerkarte "Post-Script" dazu verwendet werden, um den Aktivierungsreport zu analysieren. PREP_PROCESS_FILE wird zum Beispiel verwendet, wenn das Selektionsergebnis in eine Datei gespeichert wurde.

Beachten Sie, dass erzeugte Dateien standardmässig am Rechner, auf dem der Agent installiert ist, gepeichert werden (z.B.: R3_GET_JOB_SPOOL; FILE=).

Beispiele

Im ersten Beispiel werden alle geplanten SAP-Jobs selektiert, welche mit "FREIG" beginnen.

```
R3_GET_JOBS NAME="FREIG*",PRELIM="X"
```

Das Ergebnis wird im Aktivierungsreport ausgegeben und könnte wie diese Zeilen aussehen:

```
20000922/134303.567 - U2004943 ;FREIGABE;13450801
20000922/134303.567 - U2004943 ;FREIGABE;13455501
20000922/134303.567 - U2004943 ;FREIGABE;16221501
20000922/134303.567 - U2004943 ;FREIGABE;16225101
20000922/134303.567 - U2004943 ;FREIGABE;16234801
```

Das zweite Beispiel selektiert alle SAP-Jobs mit dem Namen "EU_REORG" und dem Zustand "Fertig". Das Ergebnis der Selektion wird in eine Datei gespeichert.

```
R3_GET_JOBS NAME='EU_REORG',FINISHED='X',FILE='jobs.txt'
```

Die ersten Zeilen in der Datei könnten beispielsweise wie diese aussehen:

```
COL=LENGTH,LENGTH_TAB='33=JOBNAME,9=JOBCOUNT'
EU_REORG                          01404301
EU_REORG                          01405401
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_GET_MONITOR

Liest Daten eines SAP-Monitors.

Transaktion: RZ20

Schnittstelle: Standard

Syntax

R3_GET_MON[ITOR]

MONITOR_SET=...
,MONITOR=...
,FILE=...

Syntaxteil	Beschreibung/Format
MONITOR_SET=	Name der Monitorsammlung (maximal 60 Zeichen) Format: Name oder Script-Literal
MONITOR=	Name eines Monitors (maximal 60 Zeichen) Format: Name oder Script-Literal
FILE=	Name einer Datei, in welcher das Ergebnis (aktuelle Monitordaten) gespeichert werden soll Format: Name oder Script-Literal
ENCODING=	Encoding für die erzeugte Ausgabedatei (Parameter FILE=). Beispiel: UTF-8 Standardwert: ISO-8859-1 Wird ein nicht unterstütztes bzw. ungültiges Encoding angegeben, so führt dies zum Abbruch des Jobs mit einer entsprechenden Fehlermeldung. In den SAP-Forms ist für dieses Feld eine Eingabehilfe verfügbar, welche alle unterstützten Encodings auflistet.

Anmerkungen

Das Script-Sprachmittel dient dazu, Daten aus dem SAP-Monitoring auszulesen und in eine Datei zu speichern.

Der SAP-Agent teilt die einzelnen Zeilen, welche ein SAP-Monitor liefert, in feste Spalten auf. Er hinterlegt Spaltennamen und Spaltenbreiten als erste Zeile in der Datei mit den aktuellen Monitordaten. Es gibt folgende Spalten:

PATH - Pfadangabe eines Wertes,
NAME -Name des Wertes,
VALUE - Aktueller Wert,
STATUS - Status: 1 = grün, 2 = gelb, 3 = rot,
DATE - Datum der Prüfung,
TIME - Zeitpunkt der Prüfung.
MESSAGE - Nachricht, die bei einem Protokollattribut angezeigt wird.

⚠ Beachten Sie, dass der Zeitstempel in UTC zurückgegeben wird und daher von Ihrer lokalen Zeit abweicht! Das Script-Sprachmittel CONV_TIMESTAMP bietet die Möglichkeit Datum und Uhrzeit in eine andere Zeitzone umzurechnen (siehe PREP_PROCESS).

Insbesondere sei auch auf die Möglichkeit hingewiesen, Monitordaten mit den Script-Sprachmitteln für Datensequenzen bereitzustellen und zu bearbeiten. Dabei wird R3_GET_MONITOR im Ereignis-Job "EVENT.R3MONITOR" aus dem Mandanten "00" verwendet. Mit der Script-Funktion PREP_PROCESS kann nun eine Datensequenz mit den Daten eines SAP-Monitors erzeugt werden.

Die Datensequenz wird durch die Script-Anweisungen :PROCESS und :ENDPROCESS verarbeitet. In Kombination mit der Script-Funktion GET_PROCESS_LINE kann auf jede einzelne Zeile der Datensequenz und deren Spalten zugegriffen werden.

ℹ Beachten Sie, dass erzeugte Dateien standardmässig am Rechner, auf dem der Agent installiert ist, gepeichert werden (z.B.: R3_GET_JOB_SPOOL; FILE=).

Siehe auch:

Allgemeines zur SAP-JCL

R3_GET_SESSIONS

Selektiert Batch-Input-Mappen und listet das Ergebnis im Aktivierungsreport oder in einer Datei auf.

Transaktion: SM35

Schnittstelle: AE

Syntax

R3_GET_SESSION[S]

NAME=...
CREDATE_FROM=...
CREDATE_TO=...
STATUS=...
[**,FILE=**...]

[**,NOFOUND=**...]
[**,ORDER_BY=**...]

Syntaxteil	Beschreibung/Format
NAME=	Selektion von Batch-Input-Mappen nach Namen Format: Script-Literal Es können die Wildcardzeichen "*" und "?" verwendet werden. "*" steht dabei für beliebige, "?" für genau ein Zeichen.
CREDATE_FROM=	Selektion von Batch-Input-Mappen nach ihrem Erstellungsdatum (Start-Datum der Selektion) Format: Script-Literal Datumsformat: JJJJMMTT
CREDATE_TO=	Selektion von Batch-Input-Mappen nach ihrem Erstellungsdatum (Ende-Datum der Selektion) Format: Script-Literal Datumsformat: JJJJMMTT
STATUS=	Selektion von Batch-Input-Mappen nach ihrem Status Format: Script-Literal Erlaubte Werte: " " (Standardwert), "R", "F" und "E" " " - Batch-Input-Mappen, welche noch zu verarbeiten sind. "R" - Batch-Input-Mappen, welche gerade verarbeitet werden (Running). "F" - Batch-Input-Mappen, deren Verarbeitung beendet ist (Finished). "E" - Batch-Input-Mappen, bei deren Verarbeitung Fehler aufgetreten sind (Error).
FILE=	Name einer Datei, in welcher das Selektionsergebnis (gefundene Batch-Input-Mappen) geschrieben werden soll Format: Script-Literal Das Selektionsergebnis wird nicht im Aktivierungsreport ausgegeben, wenn dieser Parameter verwendet wird.
NOFOUND=	Behandlung, wenn keine Batch-Input-Mappen gefunden wurden. Format des Wertes: Script-Literal Erlaubte Werte: "NORMAL" (Standardwert) und "ABEND" "NORMAL" - Das Script wird fortgesetzt, der AE Job endet normal. "ABEND" - Das Script wird nicht fortgesetzt, der AE Job endet abnormal.
ORDER_BY=	Kriterium nach der die Selektion der Mappen sortiert wird. Es können alle Feldnamen der SAP Tabelle APQI angegeben werden. Beispiel: ORDER_BY=GROUPID Der Parameter wird ab SAP-Version 4.6 unterstützt.

ENCODING=	Encoding für die erzeugte Ausgabedatei (Parameter FILE=). Beispiel: UTF-8 Standardwert: ISO-8859-1 Wird ein nicht unterstütztes bzw. ungültiges Encoding angegeben, so führt dies zum Abbruch des Jobs mit einer entsprechenden Fehlermeldung. In den SAP-Forms ist für dieses Feld eine Eingabehilfe verfügbar, welche alle unterstützten Encodings auflistet.

Beschreibung

Das Ergebnis der Selektion wird entweder in den Aktivierungsreport oder in eine Datei geschrieben. Für jede gefundene Batch-Input-Mappe gibt es eine Zeile. Der Aufbau der Zeilen im Aktivierungsreport und in der Datei ist unterschiedlich.

In der Datei werden Spalten verwendet. Die erste Zeile der Datei enthält Informationen zur Breite und zum Inhalt der Spalten. Laut dieser Festlegung sind die einzelnen Zeilen unterteilt. Die ersten 13 Stellen enthalten jeweils den Namen der Batch-Input-Mappe, weitere 21 Stellen die Queue-ID. In 13 weiteren Stellen wird der Benutzer ausgegeben, der die Batch-Input-Mappe erzeugt hat.

Mit dem Parameter ORDER_BY kann die Reihenfolge der Batch-Input-Mappen nach einem Kriterium sortiert werden.

Im Aktivierungsreport sind die Informationen innerhalb einer Zeile durch Semikolon getrennt. Zusätzlich wird ein Zeitstempel und eine AE Meldungsnummer am Anfang der Zeile ausgegeben.

Das Ergebnis der Selektion kann mit Script-Funktionen analysiert werden. PREP_PROCESS_REPORT kann in der Registerkarte "Post-Script" dazu verwendet werden, um den Aktivierungsreport zu analysieren. PREP_PROCESS_FILE wird zum Beispiel verwendet, wenn das Selektionsergebnis in eine Datei gespeichert wurde.

Mit AE-Version 2.63E wird der Erzeuger der Batch-Input-Mappen zusätzlich am Ende jeder Zeile des Aktivierungsreports oder der Datei ausgegeben. Das kann unter Umständen zu Problemen in bestehenden AE Scripts führen, wenn die Datensequenz, welche mit PREP_PROCESS_* erzeugt wurde, nicht in Spalten unterteilt ist. Das ist beispielsweise der Fall, wenn die gesamte Zeile mit String-Funktionen bearbeitet wird.

Beachten Sie, dass erzeugte Dateien standardmässig am Rechner, auf dem der Agent installiert ist, gepeichert werden (z.B.: R3_GET_JOB_SPOOL; FILE=).

Beispiele

Im ersten Beispiel werden alle Batch-Input-Mappen mit dem Namen "NI" selektiert, die heute noch zu verarbeiten sind.

```
:SET &HEUTE# = SYS_DATE(JJJJMMTT)
R3_GET_SESSIONS NAME='NI',CREDATE_FROM='&HEUTE#',CREDATE_
TO='&HEUTE#',STATUS=" "
```

Im Aktivierungsreport werden Zeilen wie diese ausgegeben:

```
20020313/135601.000 - U2004943 ;AE_TEST;20020312NI;NI
```

Das zweite Beispiel selektiert alle Batch-Input-Mappen zwischen dem 1.1.2000 und dem 1.1.2002, bei deren Verarbeitung Fehler aufgetreten sind. Das Ergebnis der Selektion wird in eine Datei gespeichert.

```
R3_GET_SESSIONS NAME='*',CREDATE_FROM='20000101',CREDATE_
TO='20020101',STATUS='E',FILE='sessions.txt'
```

Die ersten Zeilen in der Datei könnten beispielsweise wie diese aussehen:

```
COL=LENGTH,LENGTH_TAB='13=GROUPID,21=QID,13=CREATOR'
AE_TEST      20020314095728031322 NI
AE_TEST      20020314095823023148 NI
AE_TEST      20020314100932031323 NI
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_GET_SPOOLREQUESTS

Selektiert Spoolaufträge mit vordefinierten Filtern.

Transaktion: SP01

Schnittstelle: UC4

Syntax

R3_GET_SPOOLREQ[UESTS]

[**WITHOUT=**...]
[**,PROCESSING=**...]
[**,SUCCESSFUL=**...]
[**,PROBLEM=**...]
[**,OWNER=**...]
[**,AUTHORITY=**...]
[**,NAME=**...]
[**,SUFFIX1=**...]
[**,SUFFIX2=**...]
[**,DESTINATION=**...]
[**,SPOOLNR_FROM=**...]
[**,SPOOLNR_TO=**...]
[**,CREDATE_FROM=**...]
[**,CREDATE_TO=**...]
[**,TITLE=**...]
[**,RECEIVER=**...]
[**,DEPARTMENT=**...]
[**,NOFOUND=**...]

Syntaxteil	Beschreibung/Format

WITHOUT=	Selektion nach dem Auftragsstatus "ohne" Format des Wertes: Script-Literal Erlaubte Werte: "Y" (Standardwert) und "N" "Y" - Selektion wird durchgeführt "N" - Es erfolgt keine Selektion nach dem Auftragsstatus "ohne".
PROCESSING=	Selektion nach dem Auftragsstatus "in Arbeit" Format des Wertes: Script-Literal Erlaubte Werte: "Y" (Standardwert) und "N" "Y" = Selektion wird durchgeführt "N" = Es erfolgt keine Selektion nach dem Auftragsstatus "in Arbeit".
SUCCESSFUL=	Selektion nach dem Auftragsstatus "erfolgreich" Format des Wertes: Script-Literal Erlaubte Werte: "Y" (Standardwert) und "N" "Y" = Selektion wird durchgeführt "N" = Es erfolgt keine Selektion nach dem Auftragsstatus "erfolgreich".
PROBLEM=	Selektion nach dem Auftragsstatus "problematisch". Format des Wertes: Script-Literal Erlaubte Werte: "Y" (Standardwert) und "N" "Y" = Selektion wird durchgeführt "N" = Es erfolgt keine Selektion nach dem Auftragsstatus "problematisch".
OWNER=	Eigentümer des Spoolauftrags Format des Wertes: Script-Literal
AUTHORITY=	Spoolberechtigung Format des Wertes: Script-Literal
NAME=	Name des Spoolauftrags Format des Wertes: Script-Literal
SUFFIX1=	Suffix 1 des Spoolauftrags Format des Wertes: Script-Literal
SUFFIX2=	Suffix 2 des Spoolauftrags Format des Wertes: Script-Literal
DEST[INATION]=	Ausgabegerät Format des Wertes: Script-Literal
SPOOLNR_FROM=	ab Spool-Nummer Format des Wertes: Script-Literal
SPOOLNR_TO=	bis Spool-Nummer Format des Wertes: Script-Literal
CREDATE_FROM=	Erzeugungsdatum des Spoolauftrags (ab Datum) Format des Wertes: Script-Literal Format: "YYYYMMDD" Standardwert: "20010101"

CREDATE_TO=	Erzeugungsdatum des Spoolauftrags (bis Datum) Format des Wertes: Script-Literal Format: "YYYYMMDD" Standardwert: aktuelles Datum
TITLE=	Titel des Spoolauftrags Format des Wertes: Script-Literal
RECEIVER=	Empfänger der Spool-Liste Format des Wertes: Script-Literal
DEPART[MENT]=	Empfänger der Spool-Liste (Abteilung) Format des Wertes: Script-Literal
NOFOUND=	Behandlung, wenn keine Spoolaufträge gefunden werden. Format des Wertes: Script-Literal Erlaubte Werte: "NORMAL" (Standardwert) und "ABEND" "NORMAL" - Der AE-Job läuft weiter. "ABEND" - der AE-Job endet abnormal.

Beschreibung

Die Funktion schreibt am Beginn jeder Reportzeile ein ";".

Beispiel

```
R3_GET_SPOOLREQUESTS SPOOLNR_FROM='1234',SPOOLNR_
TO='1236',DESTINATION='PRNT',RECEIVER='MEIER'
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_GET_SYSTEMLOG

Liest das Systemlog eines SAP-Systems über einen vorgegebenen Zeitraum.

Transaktion: SM21

Schnittstelle: Standard

Syntax

R3_GET_SYSTEMLOG

FILE=...
[**,SERVER=**...]
[**,FROM_DATE=**...]
[**,FROM_TIME=**...]
[**,TO_DATE=**...]
[**,TO_TIME=**...]

Syntaxteil	Beschreibung/Format
FILE=	Name einer Datei, in welcher das gelesene SAP-Systemlog gespeichert werden soll. Format: Name oder Script-Literal
SERVER=	Name eines SAP-Applikationsservers Format: Name oder Script-Literal Der Name eines SAP-Applikationsservers muss in der Form *Host_SID_SYSNR* angegeben werden: *Host* = Rechnername *SID* = System-ID von SAP *SYSNR* = Nummer der SAP-Instanz
FROM_DATE=	Start-Datum für die Selektion aus dem Systemlog. Format: JJJJMMTT Standardwert: "20010101"
FROM_TIME=	Start-Zeit für die Selektion aus dem Systemlog. Format: HHMMSS Standardwert: "000000"
TO_DATE=	Ende-Datum für die Selektion aus dem Systemlog. Format: JJJJMMTT Standardwert: aktuelles Datum
TO_TIME=	Ende-Zeit für die Selektion aus dem Systemlog. Format: HHMMSS Standardwert: "235959"
ENCODING=	Encoding für die erzeugte Ausgabedatei (Parameter FILE=). Beispiel: UTF-8 Standardwert: ISO-8859-1 Wird ein nicht unterstütztes bzw. ungültiges Encoding angegeben, so führt dies zum Abbruch des Jobs mit einer entsprechenden Fehlermeldung. In den SAP-Forms ist für dieses Feld eine Eingabehilfe verfügbar, welche alle unterstützten Encodings auflistet.

Anmerkungen

Das Script-Sprachmittel dient dazu, das Systemlog eines SAP-Systems in Textform auszulesen und in eine Datei zu speichern. Wird der Name eines Applikationsservers angegeben, liefert das Script-Sprachmittel dessen Systemlog. Ohne den Namen eines Applikationsservers wird das zentrale SAP-Systemlog gelesen.

Insbesondere sei auch auf die Möglichkeit hingewiesen, das Systemlog mit den Script-Sprachmitteln für Datensequenzen bereitzustellen und zu bearbeiten. Dabei wird R3_GET_SYSTEMLOG im Event-Job "EVENT.R3SYSLOG" aus dem Mandanten "00" verwendet. Mit der Script-Funktion PREP_PROCESS kann nun eine Datensequenz mit den Daten des Systemlogs erzeugt werden.

Die Datensequenz wird durch die Script-Anweisungen :PROCESS und :ENDPROCESS verarbeitet. In Kombination mit der Script-Funktion GET_PROCESS_LINE kann auf jede einzelne Zeile der Datensequenz und deren Spalten zugegriffen werden.

Beachten Sie, dass erzeugte Dateien standardmässig am Rechner, auf dem der Agent installiert ist, gepeichert werden (z.B.: R3_GET_JOB_SPOOL; FILE=).

Beispiel

Im Beispiel wird das zentrale SAP-Systemlog der vergangenen Woche gelesen. Das Start-Datum für die Selektion aus dem Systemlog wird mit der Script-Funktion SUB_DAYS errechnet. Dabei wird vom aktuellen Tagesdatum ausgegangen.

```
:RSET &HEUTE#           = SYS_DATE ('JJJJMMTT')
:RSET &LETZTE_WOCHE#    = SUB_DAYS ('JJJJMMTT:&HEUTE#', 7)
R3_GET_SYSTEMLOG FILE = 'c:\t46_systemlog.txt', SERVER=, FROM_
DATE='&LETZTE_WOCHE#'
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_GET_VARIANTS

Listet alle verfügbaren Varianten im Aktivierungslog auf.

Transaktion: SA38

Schnittstelle: Standard

Syntax

R3_GET_VARIANT[S]

REPORT=...
[**,ERROR=**...]
[**,SELECT_OPTION=**...]

Syntaxteil	Beschreibung/Format
REP[ORT]=	Name des ABAP-Programms Format: Script-Literal
ERROR=	Anweisung für die weitere Verarbeitung nach Auftreten eines Fehlers (z.B.: Report nicht vorhanden, kein Selektionsbild, keine Variante) Format: Script-Literal Erlaubte Werte: "ABEND" (Standardwert) und "IGNORE" "ABEND" - Das Script wird nicht fortgesetzt, der AE Job endet abnormal. "IGNORE" - Das Script wird fortgesetzt, der AE Job endet normal.

SEL[ECT_OPTION]=	Art der Varianten, die selektiert werden sollen Format: Script-Literal Erlaubte Werte: "A" (Standardwert) und "B" "A" = Varianten, die für Hintergrundverarbeitung und Dialogbetrieb zugelassen sind. "B" = Varianten, die ausschließlich für die Hintergrundverarbeitung zugelassen sind.

Beispiel

```
R3_GET_VARIANTS REP='RSPO0041'
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_GET_VARIANT_CONTENTS

Zeigt den Inhalt einer Variante an.

Transaktion: SA38

Schnittstelle: AE und Standard (XBP 2.0)

Syntax

R3_GET_VARIANT_C[ONTENTS]

REPORT=...
,VARIANT=...
[**,SELNAME=**...]
[**,KIND=**...]
[**,LOW=**...]
[**,HIGH=**...]
[**,SIGN=**...]
[**,OPTION=**...]
[**,ERROR=**...]

Syntaxteil	Beschreibung/Format
REP[ORT]=	Name des Reports Format: Script-Literal oder AE-name
VAR[IANT]=	Name der Variante Format: Script-Literal oder AE-Name
SELN[AME]=	Name des Parameters Format: Script-Literal oder AE-Name

KIND=	Art der Selektion Format: Script-Literal oder AE-Name Mögliche Werte: "S" und "P" (Standardwert) "S" - Selektionskriterium "P" - Parameter
LOW=	Wert des Parameters (LOW) Format: Script-Literal oder AE-Name
HIGH=	Wert des Parameters (HIGH) Format: Script-Literal oder AE-Name
SIGN=	Kennzeichen (Include/Exclude). Format: Script-Literal oder AE-Name Erlaubte Werte: "I" (Standardwert) und "E" "I" = Include "E" = Exclude
OPTION=	Option nach der selektiert werden soll Format: Script-Literal oder AE-Name "EQ" - gleich (Standardwert) "NE" - ungleich "GT" - größer "GE" - größer oder gleich "LT" - kleiner "LE" - kleiner oder gleich "CP" - Mustervergleich (mit Wildcardzeichen) "NP" - Muster ausschließen "BT" - Intervall (auch HIGH angeben) "NB" - außerhalb Intervall (auch HIGH angeben)
ERROR=	Behandlung, wenn die Variante oder der Inhalt nicht gefunden wurde Format: Script-Literal oder AE-Name Erlaubte Werte: "ABEND" (Standardwert) und "IGNORE" "ABEND" = Das Script wird nicht fortgesetzt, der AE Job endet abnormal. "IGNORE" = Das Script wird fortgesetzt, der AE Job endet normal.

Beispiele

Der gesamte Varianteninhalt wird Zeile für Zeile ausgegeben. Falls die Variante nicht gefunden wird, bricht der Job trotzdem nicht ab.

```
R3_GET_VARIANT_C REP='RSPO0041',VAR='STANDARD',ERROR='IGNORE'
```

Alle Einträge der Tabelle VARI_VALUES bzw. VALUETAB, deren Spalte SELNAME den Wert 'MIN_ALT' enthält, werden aufgelistet. Wird 'MIN_ALT' oder die Variante nicht gefunden, bricht der Job ab.

```
R3_GET_VARIANT_C REP='RSPO0041',VAR='STANDARD',SELNAME='MIN_
ALT',ERROR='ABEND'
```

Alle Einträge der Tabelle VARI_VALUES bzw. VALUETAB, deren Spalten SELNAME=MIN_ALT, KIND=P und LOW=10 enthalten, werden ausgegeben. Wird keine entsprechende Treffermenge gefunden, bricht der Job ab.

```
R3_GET_VARIANT_C REP='RSPO0041',VAR='STANDARD',SELNAME='MIN_
ALT',KIND='P',LOW='10',ERROR='ABEND'
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_IMPORT_CALENDAR

Importiert Kalender von SAP in das Automation Engine System.

Schnittstelle: AE und Standard

Syntax

R3_IMPORT_CALENDAR

CALENDAR_ID=...
[**,TYPE=**...]
[**,FOLDER=**...]
[**,UC4_CALE=**...]
[**,UC4_KEYWORD=**...]
[**,YEAR_FROM=**...]
[**,YEAR_TO=**...]

Syntaxteil	Beschreibung/Format
CALENDAR_ID=	ID des SAP-Kalenders, der importiert werden soll. Format: Script-Literal oder AE-name Die SAP Kalender ID besteht aus genau 2 Zeichen.
TYPE=	Typ des SAP-Kalenders Format: Script-Literal oder AE-Name Erlaubte Werte: "FACTORY" (Standardwert) oder "HOLIDAY" "FACTORY" - Typ Fabrikskalender "HOLIDAY" - Typ Feiertagskalender
FOLDER=	Ordner im Madanten, in welchem die Kalender-Objekte abgelegt werden sollen. Format: Script-Literal oder AE-Name Anzugeben ist der Ordner-Pfad ohne führenden root-Ordner im Format: /<*Unterordner*>/…/<*Ordner*> Beispiel: /JOBS/TESTJOBP Standardwert: root-Ordner des Mandanten, in welchem der Job gestartet wurde.

UC4_CALE=	Name des Kalender-Objektes Format: Script-Literal oder AE-Name Existiert das angegebene Objekt nicht, so wird es erstellt. Andernfalls wird nur der Kalenderbegriff eingetragen bzw. erneuert. Standardwert: CALE.FROM.%SID%
UC4_KEYWORD=	Name des Kalenderbegriffs für den UC4 Kalender, unter welchem die Tage des SAP-Kalenders abgelegt werden Format: Script-Literal oder AE-Name Existiert der Kalenderbegriff im Kalender-Objekt bereits, so wird dieser überschrieben. Wurden die Parameter YEAR_FROM und YEAR_TO angegeben, so werden nur die Tage gelöscht, die in diesem Zeitraum fallen. Standardwert: ID des SAP Kalenders
YEAR_FROM=	Jahr, ab dem die Einträge des SAP Kalenders importiert werden sollen. Format: Script-Literal oder AE-Name Standardwert: Keine Beschränkung
YEAR_TO=	Jahr, bis zu welchem die Einträge des SAP Kalenders importiert werden sollen. Format: Script-Literal oder AE-Name Standardwert: Keine Beschränkung

Beschreibung

Das Sprachmittel importiert einen bestimmten Kalender aus dem SAP-System in das Automation Engine System. Dabei können sowohl Fabriks- als auch Feiertagskalender angegeben werden. Die Tage des SAP-Kalenders werden in einem Begriff eines bestimmten Kalender-Objektes gespeichert.

Beispiel

Folgendes Beispiel importiert den SAP Feiertragskalender AT in das Automation Engine System und trägt die Tage in den Begriff AUSTRIA im Kalender-Objekt CALE.AT ein. Dabei werden nur Tage vom Jahr 2012 verwendet.

```
R3_IMPORT_CALENDAR CALENDAR_
ID='AT',TYPE='HOLIDAY',FOLDER='/TEST/CALE',UC4_CALE='CALE.AT',UC4_
KEYWORD='AUSTRIA',YEAR_FROM='2012',YEAR_TO='2012'
```

R3_IMPORT_JOBS

Importiert Jobs von SAP in das Automation Engine System.

Schnittstelle: AE und Standard

Syntax

R3_IMPORT_JOBS

[SINGLE_JOBS=...]
[**,WORKFLOWS=**...]
[**,JOB_NAME=**...]
[**,LOGIN_NAME=**...]
[**,WORKFLOW_NAME=**...]
[**,EXTENSION=**...]
[**,LOGINS=**...]
[**,HOST=**...]
[**,QUEUE=**...]
[**,DEPENDENCY_STATE=**...]
[**,DEPENDENCY_ELSE_ACTION=**...]
[**,DEPENDENCY_ELSE_ALARM=**...]
[**,EXT_COMMAND=**...]
[**,EXT_PROGRAM=**...]
[**,LOGIN_FOLDER=**...]
[**,JOB_FOLDER=**...]
[**,WORKFLOW_FOLDER=**...]
[**,OVERRIDE=**...]
[**,JOB_TEMPLATE=**...]
[**,WORKFLOW_TEMPLATE=**...]

Innerhalb der Werte mancher Parameter können spezielle Platzhalter eingetragen werden, die mit dem Zeichen % eingeleitet und abgeschlossen werden (zum Beispiel: %SID%). Am Standardwert der Parameter sehen Sie, ob und welche Platzhalter verwendet werden können. Eine Liste mit Erklärung der Platzhalter finden Sie unterhalb der Syntax-Tabelle.

Syntaxteil	Beschreibung/Format
SINGLE_JOBS=	Verhalten, ob für jeden Jobstep ein Job-Objekt erstellt werden soll. Format: Script-Literal oder AE-name Erlaubte Werte: "YES" (Standardwert) oder "NO" "YES" - Für jede R3_* Funktion ein eigenes Job-Objekt anlegen. "NO" - Die erstellten Job-Objekte können auch mehrere R3_* Funktionen enthalten.
WORKFLOWS=	Workflows für die importierten Jobs erstellen Format: Script-Literal oder AE-Name Erlaubte Werte: "" (Standardwert), "1" oder "2" "" - Keine Workflow-Objekte erstellen "1" - Workflows für alle importieren Jobs erstellen "2" - Es werden nur Workflows für Jobs angelegt, die mehrere Steps besitzen.
JOB_NAME=	Namensschema für die erstellten Job-Objekte Format: Script-Literal oder AE-Name Standardwert: JOBS.%SID%.%CLIENT%@%JOBNAME%

LOGINS=	Login-Objekte erstellen Erlaubte Werte: "" (Standardwert), "1" oder "2" "" - Keine Login-Objekte generieren. "1" - Login-Objekte anlegen. Im Objekt wird der Name des Agenten eingetragen. "2" - Login-Objekte anlegen. Im Objekt wird "*" anstelle des Agentennamens eingetragen.
LOGIN_NAME=	Namensschema für die importierten Login-Objekte Format: Script-Literal oder AE-Name Standardwert: LOGIN.%SID%.%CLIENT%@%USER%
WORKFLOW_ NAME=	Namensschema für die erstellten Workflow-Objekte Format: Script-Literal oder AE-Name Standardwert: JOBP.%SID%.%CLIENT%@%JOBNAME%
EXTENSION=	Erweiterung für den Objektnamen von Jobs, die pro Step erstellt werden. Format: Script-Literal oder AE-Name Dieser Parameter ist nur relevant, wenn SINGLE_JOBS auf den Wert YES gesetzt ist. Standardwert: %STEP%
HOST=	Host-Attribut der angelegten Jobs Format: Script-Literal oder AE-Name Standardwert: %SID%
QUEUE=	Queue-Attribut der angelegten Jobs Format: Script-Literal oder AE-Name Standardwert: CLIENT_QUEUE
DEPENDENCY_ STATE=	Erwarteter Status, der im Workflow in den Eigenschaften des Jobs (Abhängigkeiten) eingetragen werden soll Format: Script-Literal oder AE-Name Nur relevant, wenn Workflows erstellt werden. Standardwert: ANY_OK

DEPENDENCY_ ELSE_ACTION=	Sonst-Anweisung für die Abhängigkeiten des Jobs im Workflow (Aufgaben-Eigenschaften) Format: Script-Literal oder AE-Name Nur relevant, wenn Workflows erstellt werden. Erlaubte Werte: "" (Standardwert), "ABORT", "BLOCK", "BLOCK_ ABORT" oder "SKIP" "" - Wert des Workflow-Templates "ABORT" - Aufgabe und Workflow abbrechen "BLOCK" - Der Workflow blockiert bei der entsprechenden Aufgabe "BLOCK_ABORT" - Workflow blockiert bei der Aufgabe und sendet ein Abbruchsignal zu einem übergeordneten Workflow (wenn vorhanden) "SKIP" - Aufgabe überspringen
DEPENDENCY_ ELSE_ALARM=	Name eines Objektes, welches ausgeführt werden soll, wenn die Abhängigkeiten des Jobs im Workflow nicht erfüllt werden (Aufgaben-Eigenschaften) Format: Script-Literal oder AE-Name Nur relevant, wenn Workflows erstellt werden.
EXT_COMMAND=	Externe Kommando-Steps miteinbeziehen Format: Script-Literal oder AE-Name Erlaubte Werte: "YES" (einbeziehen) oder "NO" (nicht einbeziehen, Standardwert)
EXT_PROGRAM=	Externe Programm-Steps miteinbeziehen Format: Script-Literal oder AE-Name Erlaubte Werte: "YES" (einbeziehen) oder "NO" (nicht einbeziehen, Standardwert)
LOGIN_FOLDER=	Pfad des Ordners im Mandanten, in welchem die erstellten Login-Objekte abgelegt werden Format: Script-Literal oder AE-Name Standardwert: <Kein Ordner>
JOB_FOLDER=	Pfad des Ordners im Mandanten, in welchem die erstellten Job-Objekte abgelegt werden Format: Script-Literal oder AE-Name Standardwert: <Kein Ordner>
WORKFLOW_ FOLDER=	Pfad des Ordners im Mandanten, in welchem die erstellten Workflow-Objekte abgelegt werden Format: Script-Literal oder AE-Name Standardwert: <Kein Ordner>
OVERRIDE=	Existierende Objekte überschreiben Format: Script-Literal oder AE-Name Erlaubte Werte:"YES" (Standardwert) oder "NO" "YES" - Objekte überschreiben "NO" - Objekte mit dem selben Namen werden nicht ersetzt.

JOB_TEMPLATE=	Name eines SAP Job-Objektes, welches als Basis für die Joberstellung verwendet werden soll. Format: Script-Literal oder AE-Name Standardwert: "JOBS.SAP"
WORKFLOW_ TEMPLATE=	Name eines Workflow-Objektes (Standard-Workflow), welches als Basis für die Workflow-Erstellung verwendet werden soll. Format: Script-Literal oder AE-Name Standardwert: "JOBP"

Platzhalter für Parameter-Werte:

- **%SID%** - ID des SAP-Systems, mit dem der Agent verbunden ist.
- **%CLIENT%** - Nummer des SAP-Mandanten, an welchem der CPIC-Benutzer angemeldet ist.
- **%JOBNAME%** - Name des Jobs im SAP-System
- **%USER%** - Name des SAP-Benutzers
- **%STEP%** - Name des Job-Steps

Es spielt keine Rolle ob die Parameter in Groß- und / oder Kleinschreibung angegeben werden.

Beschreibung

Das Sprachmittel ermöglicht es, Jobs aus SAP in Form von Job-Objekten in das Automation Engine System zu übernehmen. Dabei wird das SAP-System und der SAP-Mandant verwendet, an welchem der jeweilige SAP-Agent angemeldet ist, auf welchem das Sprachmittel R3_IMPORT_ JOBS ausgeführt wird.

Vor der Funktion R3_IMPORT_JOBS ist eine Selektion der zu importierenden Jobs mit dem Sprachmittel R3_GET_JOBS erforderlich. Alle SAP-Jobs, die mit R3_GET_JOBS ausgewählt wurden, werden als Job-Objekte in das Automation Engine System geladen. Wird keine Selektion vor dem Aufruf von R3_IMPORT_JOBS durchgeführt, so kommt es zum Abbruch.

Weiters besteht die Möglichkeit, Workflow- und Login-Objekte für die Jobs zu erstellen. Dieses Verhalten ist mit den Parametern LOGIN* und WORKFLOW* steuerbar (siehe oben).

Die Jobs werden in die erstellten Workflows eingefügt und anschließend die Abhängigkeiten in den Workflow-Eigenschaften entsprechend gesetzt (Parameter DEPENDENCY*=). Workflows können entweder für alle Jobs oder für Jobs mit mehreren Steps erstellt werden.

Beispiel

In folgendem Beispiel werden alle Jobs in SAP deren Name mit "TEST" beginnt selektiert und in das Automation Engine System übernommen.

```
R3_GET_JOBS NAME="TEST*",USER="MEIER"
R3_IMPORT_JOBS JOB_NAME='JOBS.UC4.%JOBNAME%.%STEP%'
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_MODIFY_INTERCEPTION

Ändert die Filtertabelle für Intercepted Jobs.

Transaktion: -

Schnittstelle: Standard (XBP 2.0)

Syntax

R3_MODIFY_INTERC[**EPTION**]

[**,NAME=**...]
[**,USER=**...]
[**,MODE=**...]

Syntaxteil	Beschreibung/Format
NAME=	Name eines SAP-Jobs oder mehrerer SAP-Jobs (mit Wildcardzeichen *). Format des Wertes: Script-Literal Standardwert: "*"
USER=	Name eines SAP-Benutzers oder mehrerer SAP-Benutzer (mit Wildcardzeichen *). Format des Wertes: Script-Literal Standardwert: "*"
MODE=	Verarbeitungsmodus bei neuem Tabelleneintrag. Format des Wertes: Script-Literal Erlaubte Werte: "REPLACE" (Standardwert) und "APPEND" "REPLACE" - Neuer Tabelleneintrag ersetzt bestehende Tabelleneinträge. "APPEND" - Neuer Tabelleneintrag wird bestehenden Tabelleneinträgen hinzugefügt.

Anmerkungen

Das Script-Sprachmittel ändert den Inhalt der Tabelle aus dem SAP-System, in welcher die Bedingungen für Batch-Jobs definiert wurden. Dabei wird der SAP-Mandant verwendet, der im Login-Objekt, auf welches der AE Job referenziert, angegeben ist.

Wird das Script-Sprachmittel mit MODE=REPLACE verwendet, werden alle Tabelleneinträge des SAP-Mandanten gelöscht und durch den neuen Tabelleneintrag ersetzt. Bei MODE=APPEND wird der neue Tabelleneintrag zu den bestehenden Tabelleneinträge des SAP-Mandanten hinzugefügt.

⚠ Beachten Sie, dass das Script-Sprachmittel nur beim Einsatz von XBP 2.0 unterstützt wird!

Beispiel

Im Beispiel werden zunächst alle definierten Intercepted Jobs ermittelt.

```
R3_GET_INTERCEPTION
```

Das Aktivierungsprotokoll enthält die folgenden Zeilen.

```
20030325/153054.000 - U2004943 ;001;IC*;*
20030325/153054.000 - U2004943 ;001;UC4*;UC4.WG
```

Anschließend werden alle Tabelleneinträge durch einen neuen Eintrag ersetzt und die definierten Intercepted Jobs nochmals abgefragt.

```
R3_MODIFY_INTERCEPTION NAME='UC4*',USER='UC4.WG'
R3_GET_INTERCEPTION
```

Im Aktivierungsprotokoll wir nur noch eine Zeile ausgegeben.

```
20030325/153054.000 - U2004943 ;001;UC4*;UC4.WG
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_MODIFY_JOB

Modifziert einen ABAP-Step.

Transaktion: SM36

Schnittstelle: Standard

Syntax

R3_MODIFY_JOB

NAME=...
,JOBCOUNT=...
,STEP=...
,REPORT=...
[**,VAR[IANT]=**...]
[**,ARCHIVE_O[BJECT]=** ...]
[**,ARCHIVE_S[APOBJECT]=**...]
[**,ARCHIVE_I[NFO]=**...]
[**,DEST[INATION]=**...]
[**,IMM[EDIATELY]=**...]
[**,REL[EASE]=**...]
[**,COPIES=**...]
[**,ARCIVE_M[ODE]=**...]
[**,AUTHORITY=**...]
[**,SAP_COVER[_PAGE]=**...]
[**,COVER[PAGE]=**...]
[**,EXPIR[ATION]=**...]
[**,RECEIVER=**...]
[**,LINE_COUNT=**...]
[**,LINE_SIZE=**...]

Syntaxteil	Beschreibung/Format

NAME=	Name des SAP-Jobs Format: Script-Literal oder AE-name In Verbindung mit der Jobnummer wird der einzelne SAP-Job eindeutig identifiziert.
JOBCOUNT=	Nummer des SAP-Jobs Format: Script-Literal In Verbindung mit dem Namen wird ein SAP-Job eindeutig identifiziert.
STEP=	Nummer des ABAP-Steps, der modifiziert werden soll Format: Zahl ohne Hochkommata
RE[PORT]=	Name des Reports Format: Script-Literal oder AE-Name
Parameter	
VAR[IANT]=	Name der Variante Format: Script-Literal oder AE-Name
ARCHIVE_O [BJECT]=	Archivierungsparameter: Dokumentart des Archivobjektes Format: Script-Literal oder AE-Name
ARCHIVE_S [APOBJECT]=	Archivierungsparameter: Objekttyp des SAP-Objektes Format: Script-Literal oder AE-Name
ARCHIVE_I[NFO]=	Archivierungsparameter: Infofeld Format: Script-Literal oder AE-Name
DEST[INATION]=	Druckparameter: Ausgabegerät Format: Script-Literal oder AE-Name
IMM[EDIATELY]=	Druckparameter: Sofort ausgeben Format: Script-Literal oder AE-Name Erlaubte Werte: "YES" und "NO" (Standardwert)
REL[EASE]=	Druckparameter: Löschen nach Ausgabe Format: Script-Literal oder AE-Name Erlaubte Werte: "YES" und "NO" (Standardwert)
COPIES=	Druckparameter: Anzahl Ausdrucke Format: Zahl ohne Hochkommata Standardwert: "0"
ARCHIVE_M[ODE]=	Druckparameter: Archivierungsmodus Format: Script-Literal oder AE-Name Erlaubte Werte: "1" (Standardwert), "2" und "3" "1" - Nur Drucken "2" - Nur Archivieren "3" - Drucken und Archivieren
AUTHORITY=	Druckparameter: Berechtigung Format: Script-Literal oder AE-Name

SAP_COVER[_PAGE]=	Druckparameter: SAP-Deckblatt Format: Script-Literal oder AE-Name Erlaubte Werte: "" (Standardwert), "X" und "D" "" - Kein Deckblatt ausgeben "X" - Deckblatt ausgeben "D" - Deckblattausgabe hängt von der Einstellung des Druckers ab
COVER[PAGE]=	Druckparameter: Selektions-Deckblatt Format: Script-Literal oder AE-Name Erlaubte Werte: "YES" und "NO" (Standardwert) "YES" - Selektions-Deckblatt drucken "NO" - Kein Selektions-Deckblatt drucken
EXPIR[ATION]=	Druckparameter: Spool-Verweildauer Format: Zahl ohne Hochkommata Standardwert: "0"
RECEIVER=	Druckparameter: Empfänger Format: Script-Literal oder AE-Name
LINE_COUNT=	Druckparameter: Anzahl Listzeilen Format: Zahl ohne Hochkommata Standardwert: "0"
LINE_SIZE=	Druckparameter: Zeilenbreite der Liste. Format: Zahl ohne Hochkommata Standardwert: "0"

Beschreibung

Das Script-Sprachmittel ermöglicht es, einen ABAP-Step zu modifizieren. Nachdem der SAP-Job selektiert wurde, können für einen ABAP-Step der Name des Reports, der Name der Variante und verschiedene Archivierungs- und Druckparameter neu gesetzt werden.

Die Parameter entsprechen den Feldern der SAP-Dictionary Struktur PRI_PARAMS und ARC_PARAMS. Verwenden Sie bitte Dictionary oder BAPI-Browser, um detailliertere Informationen zu den einzelnen Feldern zu erhalten.

Beispiel

Im Beispiel wird der SAP-Job mit dem Namen "MYJOB" und der Nummer "13541601" modifiziert. Als Step "2" wird "RSM04000" als durchzuführendes ABAP-Programm eingetragen.

```
R3_MODIFY_JOB NAME="MYJOB",JOBCOUNT=13541601,STEP=2,REPORT=RSM04000
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_MODIFY_VARIANT

Modifiziert einen Eintrag in einer Variante.

Transaktion: SA38

Schnittstelle: AE und Standard (XBP 2.0)

Syntax (Parameter)

R3_MODIFY_VARIANT

REPORT=...
,VARIANT=...
,SELNAME=...
,KIND=P
,LOW=...
[**,VERIFY=**...]
[**,DELAY=**...]
[**,MERGE=**...]

Syntaxteil	Beschreibung/Format
REP[ORT]=	Name des Reports. Format: Name oder Script-Literal
VAR[IANT]=	Name der Variante. Format: Name oder Script-Literal
SELN[AME]=	Name des Parameters. Format: Name oder Script-Literal
KIND=P	Typ des Parameters = Parameter. Format: Name oder Script-Literal
LOW=	Wert des Parameters. Format: Name oder Script-Literal
VERIFY=	Modifizieren der Variante überprüfen. Format: Name oder Script-Literal Erlaubte Werte: "YES" und "NO" (Standardwert) "YES" = Modifizierte Variante überprüfen. "NO" = Modifizierte Variante nicht überprüfen.
DELAY=	Zeitspanne in Sekunden, die der Agent nach dem Modifizieren einer Variante wartet. Format: Zahl Standardwert: "0" Nach dieser Zeitspanne sollte die Buffer-Synchronisation zwischen Applikationsservern abgeschlossen sein, wenn das SAP-System mit mehreren Applikationsservern betrieben wird.

MERGE=	Parameter zusammenmischen Erlaubte Werte: "YES" (Standardwert) oder "NO" "YES" = Varianten-Parameter zusammenmischen "NO" = Intialwerte der Varianten-Parameter verwenden.

Syntax (Select-Optionen)

R3_MODIFY_VARIANT

REPORT=...
,VARIANT=...
,SELNAME=...
,KIND=S
,LOW=...
[**,HIGH=**...]
[**,SIGN=**...]
[**,OPTION=**...]
[**,MODE=**...]
[**,VERIFY=**...]
[**,DELAY=**...]
[**,MERGE=**...]

Syntaxteil	Beschreibung/Format
REP[ORT]=	Name des Reports. Format: Name oder Script-Literal
VAR[IANT]=	Name der Variante. Format: Name oder Script-Literal
SELN[AME]=	Name der Select-Option. Format: Name oder Script-Literal
KIND=S	Typ des Parameters = Select-Option Format: Name oder Script-Literal
LOW=	Wert des Parameter (LOW). Format: Name oder Script-Literal
HIGH=	Wert des Parameters (HIGH) Format: Name oder Script-Literal Nur bei OPTION=BT oder NB (Intervall) angeben!
SIGN=	Kennzeichen (Include/Exclude). Format: Name oder Script-Literal Erlaubte Werte: "I" (Standardwert) und "E" "I" = Include "E" = Exclude

OPTION=	Option, wie selektiert werden soll. Format: Name oder Script-Literal "EQ" - gleich (Standardwert) "NE" - ungleich "GT" - größer "GE" - größer oder gleich "LT" - kleiner "LE" - kleiner oder gleich "CP" - Mustervergleich (mit Wildcardzeichen) "NP" - Muster ausschließen "BT" - Intervall (auch HIGH angeben) "NB" - außerhalb Intervall (auch HIGH angeben)
MODE=	Selektionsbegriff ändern oder hinzufügen. Format: Name oder Script-Literal Erlaubte Werte: "REPLACE" (Standardwert) und "APPEND" Damit ist es möglich, Mehrfachselektionen zu erstellen. Bei Angabe von MODE=APPEND ist der Parameter MERGE= automatisch immer YES.
VERIFY=	Modifizieren der Variante überprüfen. Format: Name oder Script-Literal Erlaubte Werte: "YES" und "NO" (Standardwert) "YES" - Modifizierte Variante überprüfen. "NO" - Modifizierte Variante nicht überprüfen.
DELAY=	Zeitspanne in Sekunden, die der Agent nach dem Modifizieren einer Variante wartet. Format: Zahl Standardwert: "0" Nach dieser Zeitspanne sollte die Buffer-Synchronisation zwischen Applikationsservern abgeschlossen sein, wenn das SAP-System mit mehreren Applikationsservern betrieben wird.
MERGE=	Parameter zusammenmischen Erlaubte Werte: "YES" (Standardwert) oder "NO" "YES" = Varianten-Parameter zusammenmischen "NO" = Intialwerte der Varianten-Parameter verwenden.

Anmerkungen

Je nach Eintrag in der Variante ist zu unterscheiden:

- Parameter: werden im Report mit PARAMETERS abgefragt und
- Select-Optionen: werden im Report mit SELECT-OPTIONS abgefragt.

Hinweise zum Modifizieren von Varianten:

- Nicht angegebene Einträge bleiben unverändert.
- Erfolgt für eine Select-Option eine Änderung, so kann der Wertebereich maximal durch eine neue Änderung eine andere Gültigkeit erhalten. Ein Rücksetzen auf Initialwert ist nicht möglich.

- Freie Abgrenzungen einer Variante (bei logischen Datenbanken) können hiermit nicht geändert werden.
- Werden für eine Select-Option mehrere Einzelwerte übergeben, so entspricht die Reihenfolge in der Variante letztendlich nicht unbedingt der angegebenen Reihenfolge!
- Verwendet das ABAP-Programm, zu welchem die Variante gehört, eine logische Datenbank, kann das Modifizieren der Variante zu Fehlverhalten im SAP System führen (beispielsweise bei logischer Datenbank PSJ). Dies liegt daran, dass das Selektionsbild der logischen Datenbank im SAP System dynamisch modifiziert wird. In solchen Fällen empfiehlt sich, eine Original-Variante zu kopieren und die Kopie zu ändern.

⚠ Bei einer SAP Basis Version kleiner 7.10 und / oder einer XBP-Version kleiner 3.0 sind die Werte der Parameter und Selektions-Optionen (auch Intervalle von/bis) auf eine Maximallänge von 45 Zeichen beschränkt!

Beispiel (Parameter)

```
R3_MODIFY_VARIANT REPORT=RSUSR002,VAR=SAP_
STANDARD,SELN=TCODE,KIND=P,LOW='SE01'
```

Beispiele (Select-Optionen)

Alle Benutzer selektieren, deren Name mit "B" beginnt.

```
R3_MODIFY_VARIANT REP=RSUSR002,VAR=SAP_
STANDARD,SELN=USER,KIND=S,LOW='B*',SIGN=I,OPTION=CP
```

Alle Benutzer selektieren, außer jenen, deren Name zwischen "USER100" und "USER199" liegt.

```
R3_MODIFY_VARIANT
...,SELN=USER,KIND=S,LOW='USER100',HIGH='USER199',SIGN=I,OPTION=NB
```

Buchungskreis 10 und 71-77 selektieren - Mehrfachselektion.

```
R3_MODIFY_VARIANT
REP=RF...,VAR=v1,SELN=BUKRS,KIND=S,LOW='10',SIGN=I,OPTION=EQ,MODE=R
```

```
R3_MODIFY_VARIANT
REP=RF...,VAR=v1,SELN=BUKRS,KIND=S,LOW='71',HIGH='77',SIGN=I,OPTION=BT
,MODE=A
```

Siehe auch:

Allgemeines zur SAP-JCL
Beispielsammlung
Aufruf eines MBeans

R3_RAISE_EVENT

Löst ein in SAP definiertes Ereignis aus.

Transaktion: SM64

Schnittstelle: AE und Standard (XBP 2.0)

Syntax

R3_RAISE_EVENT

ID=...
[,**PARAM**=...]
[,**TARGET_SERVER**=...]

Syntaxteil	Beschreibung/Format	AE-Schnittstelle erforderlich
ID=	Name des Ereignisses Format: Script-Literal Erlaubt ist eine Zeichenkette mit maximal 32 Zeichen.	
PARAM=	Parameter für das Ereignis Format: Script-Literal Erlaubt ist eine Zeichenkette mit maximal 64 Zeichen.	
TARGET_SERVER=	Zielsystem, in welchem das Ereignis ausgelöst werden soll Format: Script-Literal Standardwert: Inhalt des Feldes "Zielsystem" der Registerkarte "Host-Attribute".	✓

Anmerkungen

Mit dem Script-Sprachmittel kann ein Ereignis ausgelöst werden, welches in SAP mit der Transaktion SM62 definiert wurde. Dabei kann es sich um ein Systemereignis - ein von SAP vordefiniertes Ereignis - oder um ein Benutzerereignis handeln. Als Ereignis-Parameter kann eine Zeichenkette mit beliebigem Inhalt angegeben werden. Wird dem Script-Sprachmittel kein Zielsystem, in welchem das Ereignis ausgelöst werden soll, übergeben, so wird das in der Registerkarte **Host-Attribute** eingetragene Zielsystem verwendet.

Beispiel

Im Beispiel soll das Ereignis "Test" ausgelöst werden. Zusätzlich wird der Ereignis-Parameter "Myparam" angegeben.

```
R3_RAISE_EVENT ID="TEST",PARAM="Myparam"
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_SCHEDULE_JOB_CANCEL

Setzt einen bereits freigegebenen SAP-Job auf den Status "Geplant" zurück.

Transaktion: SM37

Schnittstelle: AE

Syntax

R3_SCHEDULE_JOB_CANCEL

NAME=...
,JOBCOUNT=...

Syntaxteil	Beschreibung/Format
NAME=	Name des SAP-Jobs. Format: Script-Literal oder AE-name In Verbindung mit *Jobnummer* wird der einzelne SAP-Job eindeutig identifiziert.
JOBCOUNT=	Nummer des SAP-Jobs. Format: Script-Literal In Verbindung mit *Jobname* wird der SAP-Job eindeutig identifiziert.

Beschreibung

Mit diesem Script-Sprachmittel kann die Freigabe eines SAP-Jobs zurückgenommen werden. Der Status des SAP-Jobs wird von "Freigegeben" auf "Geplant" geändert.

SAP-Jobs mit dem Status "Freigegeben" können mit R3_GET_JOBS ermittelt werden. Das Ergebnis wird im Aktivierungsreport aufgelistet und kann in der Registerkarte **Post-Script** analysiert und verarbeitet werden.

Beispiel

Im Beispiel wird der SAP-Job mit dem Namen "MYJOB" und der Nummer "13541601" zurückgesetzt. Sein Status wird von "Freigegeben" auf "Geplant" geändert.

```
R3_SCHEDULE_JOB_CANCEL NAME="MYJOB",JOBCOUNT=13541601
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_SEND_SPOOL_REQUEST

Versendet einen existierenden Spoolauftrag.

Transaktion: SP01

Schnittstelle: UC4

Syntax

R3_SEND_SPOOL_REQ[UEST]

[**JOBNAME=**...]
[**,JOBCOUNT=**...]
[**,STEP=**...]

[**,SPOOLNR=**...]
[**,RECIPIENT=**...]
[**,ADDRESSTYPE=**...]
[**,EXPRESS=**...]
[**,COPY=**...]
[**,BLINDCOPY=**...]
[**,NOFORWARD=**...]
[**,LINE_FROM=**...]
[**,LINE_TO=**...]
[**,TITLE=**...]
[**,ERROR=**...]
[**,NOPRINT=**...]
[**,DELIVER=**...]
[**,STATUSBYMAIL=**...]

Syntaxteil	Beschreibung/Format
JOBNAME=	Name eines Hintergrund-Jobs Format des Wertes: Script-Literal
JOBCOUNT=	Nummer des Hintergrund-Jobs Format des Wertes: Script-Literal
STEP=	Step-Nummer Format des Wertes: Zahl
SPOOLNR=	Spool-Listen-Nummer Format des Wertes: Script-Literal Dieser Parameter ist eine Alternative zur Angabe von JOBNAME=, JOBCOUNT= und STEP=.
RECIPIENT=	Empfänger Format des Wertes: Script-Literal
ADDRESSTYPE=	Adresstyp Format des Wertes: Script-Literal Erlaubte Werte: "B" (Standardwert) - SAP Benutzer "C" - Allgemeine Verteilerliste "D" - X.500 Adresse "F" - Faxnummer "G" - Organisationsobjekt/ID "H" - Organisationseinheit/Planstelle "J" - SAP Objekt "L" - Telexnummer "O" - SAPoffice Benutzer "P" - Persönliche Verteilerliste "R" - SAP Benutzer in einem anderen SAP System "U" - Internet Adresse "X" - X.400 Adresse

EXPRESS=	Sende Expressnachricht Format des Wertes: Script-Literal Erlaubte Werte: "" (Standardwert) und "X" "" - Es erfolgt keine Versendung. "X" - Expressnachricht wird geschickt.
COPY=	Sende als Kopie Format des Wertes: Script-Literal Erlaubte Werte: "" (Standardwert) und "X" "" - Es erfolgt keine Versendung. "X" - Kopie wird geschickt.
BLINDCOPY=	Sende als geheime Kopie. Format des Wertes: Script-Literal Erlaubte Werte: "" (Standardwert) und "X" "" - Es erfolgt keine Versendung. "X" - Geheime Kopie wird geschickt.
NOFORWARD=	Kein Weiterleiten erlaubt Format des Wertes: Script-Literal Erlaubte Werte: "" (Standardwert) und "X" "" - Das Weiterleiten ist erlaubt. "X" - Das Weiterleiten ist nicht gestattet.
LINE_FROM=	ab Linie Format des Wertes: Zahl Standardwert: "0"
LINE_TO=	bis Linie Format des Wertes: Zahl Standardwert: "0"
TITLE=	Mail-Titel (Betreff) Format des Wertes: Script-Literal
ERROR=	Behandlung, wenn ein Fehler auftritt. Format des Wertes: Script-Literal Erlaubte Werte: "IGNORE" und "ABEND" (Standardwert) "IGNORE" - Der AE-Job läuft weiter. "ABEND" - der AE-Job endet abnormal
NOPRINT=	Kein Drucken erlaubt Format des Wertes: Script-Literal Erlaubte Werte: "Y" und "N" (Standardwert) "Y" - Das Drucken ist nicht gestattet. "N" - Das Drucken ist erlaubt.

DELIVER=	Empfangsquittung Format des Wertes: Script-Literal Erlaubte Werte: "" (Standardwert), "A", "E" und "N" "" - Wie im SAP-System eingestellt "A" - Immer "E" - Nur im Fehlerfall "N" - Nie
STATUSBYMAIL=	Empfangsquittung per Mail Format des Wertes: Script-Literal Erlaubte Werte: "" (Standardwert), "A", "E" und "N" "" - Wie im SAP-System eingestellt "A" - Immer "E" - Nur im Fehlerfall "N" - Nie

Anmerkungen

Die Funktion kann auf drei Arten verwendet werden:

1. Entweder Sie definieren des Spool mit den Parametern JOBNAME= und JOBCOUNT=. Der Parameter STEP= ist optional, Standardwert ist 1.
2. Sie geben direkt die SPOOLNR= an.
3. Oder Sie verwenden keinen dieser Parameter. In diesem Fall wird das Spool aus jenem Job in SAP bezogen, der zuletzt über den AE-Job durchgeführt wurde. Die Angabe einer Step-Nummer ist möglich.

⚠ Die Verwendung der Parameter NOPRINT=, DELIVER= und STATUSBYMAIL= setzt die folgenden SAP Supportpackages voraus:

- für 4.6C: SAPKB46C52
- für 6.20: SAPKB62059
- für 6.40: SAPKB64017
- für 7.00: SAPKB70008

Beispiel

```
R3_SEND_SPOOL_REQUEST
JOBNAME='MYJOB',JOBCOUNT=123,STEP=5094,RECIPIENT='Meier',ADDRESSTYPE='
O'
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_SET_BDCDATA

Definiert BDC-Daten.

Transaktion: -

Schnittstelle: UC4

Syntax

R3_SET_BDCDATA

PROGRAM=...
,DYNPRO=...
[**,DYBEGIN=**...]

Syntaxteil	Beschreibung/Format
PROGRAM=	Name des Programms, maximal 40 Zeichen. Format: Script-Literal
DYNPRO=	Nummer der Bildschirmmaske, vierstellig. Format: Script-Literal
DYBEGIN=	Kennzeichen, ob eine neue Bildschirmmaske beginnt. Format: Script-Literal Erlaubte Werte: "" (Standardwert), "X" "" = Keine neue Bildschirmmaske. "X" = Neue Bildschirmmaske.

R3_SET_BDCDATA

,FNAM=...
,FVAL=...

Syntaxteil	Beschreibung/Format
FNAM=	Name des Feldes, maximal 132 Zeichen. Format: Script-Literal
FVAL=	Wert, der dem Feld zugewiesen werden soll, maximal 132 Zeichen. Format: Script-Literal

Anmerkungen

Das Script-Sprachmittel dient der Definition von BDC-Daten (Batch Data Communication). Es erlaubt damit, Feldern von Bildschirmmasken im Batchmodus Werte zuzuweisen. Der SAP-Agent speichert diese Zuweisungen zunächst in einer internen Tabelle. R3_CALL_TRANSACTION leitet die eigentliche Verarbeitung ein. Die interne Tabelle wird anschließend zurückgesetzt.

R3_SET_BDCDATA wird zunächst mit den Parametern PROGRAM=, DYNPRO= und DYBEGIN= aufgerufen. Danach werden mit den Parametern FNAM= und FVAL= die gewünschten Felder mit Werten versorgt. Die Definition von BDC-Daten kann beliebig oft für die gleiche oder eine andere Bildschirmmaske wiederholt werden.

Die Namen der Bildschirmmasken und der einzelnen Felder können online im SAP-System ermittelt werden. Dazu kann die Transaktion über das Menü *System - Status* gestartet und die technische Information mit der F1-Hilfetaste aufgerufen werden. Auch die Aufzeichnungsroutine der Transaktion "SM35" liefert die Namen der Bildschirmmasken und Felder.

Beispiel

Im Beispiel werden die Bildschirmmasken der gesamten Transaktion "SA38" mit Daten gefüllt. Die Bildschirmmasken werden in der vorgegebenen Reihenfolge aufgerufen und die Felder mit Werten versorgt. Anschließend wird die Transaktion "SA38" gestartet, um ein Update der Daten durchzuführen.

```
R3_SET_BDCDATA PROGRAM="SAPMS38M", DYNPRO="0101", DYBEGIN="X"
R3_SET_BDCDATA FNAM="BDC_OKCODE", FVAL="=SSET"
R3_SET_BDCDATA FNAM="RS38M-PROGRAMM", FVAL="RSEINB00"
R3_SET_BDCDATA PROGRAM="SAPLSVAR", DYNPRO="0302", DYBEGIN="X"
R3_SET_BDCDATA FNAM="BDC_OKCODE", FVAL="=CHNG"
R3_SET_BDCDATA FNAM="RSVAR-VARIANT", FVAL="UM-V1"
R3_SET_BDCDATA FNAM="RSVAR-FLAG1", FVAL="X"
R3_SET_BDCDATA PROGRAM="RSEINB00", DYNPRO="1000", DYBEGIN="X"
R3_SET_BDCDATA FNAM="BDC_OKCODE", FVAL="=SAVE"
R3_SET_BDCDATA FNAM="P_FILE", FVAL="test.txt"
R3_SET_BDCDATA PROGRAM="RSEINB00", DYNPRO="1000", DYBEGIN="X"
R3_SET_BDCDATA FNAM="BDC_OKCODE", FVAL="=VBAC"
R3_SET_BDCDATA PROGRAM="SAPLSVAR", DYNPRO="0302", DYBEGIN="X"
R3_SET_BDCDATA FNAM="BDC_OKCODE", FVAL="/EBACK"
R3_SET_BDCDATA PROGRAM="SAPMS38M", DYNPRO="0101", DYBEGIN="X"
R3_SET_BDCDATA FNAM="BDC_OKCODE", FVAL="=BACK"
R3_CALL_TRANSACTION CODE="SA38", UPDATE="S"
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_SET_FREE_SELECTION

Definiert eine freie Abgrenzung.

Transaktion: SA38

Schnittstelle: Standard (XBP 3.0)

Syntax

R3_SET_FREE_SELECTION

,TABLENAME=...
,FIELDNAME=...
,LOW=...
[**,HIGH=**...]
[**,SIGN=**...]
[**,OPTION=**...]

Syntaxteil	Beschreibung/Format
TABLENAME=	Name der Tabelle Format: Script-Literal
FIELDNAME=	Name des Feldes Format: Script-Literal

LOW=	Wert des Parameters (LOW) Format: Script-Literal
HIGH=	Wert des Parameters (HIGH) Format: Script-Literal
SIGN=	Kennzeichen (Include/Exclude) Format: Script-Literal Erlaubte Werte: "I" (Standardwert) und "E" "I" = Include "E" = Exclude
OPTION=	Option nach der selektiert werden soll Format: Script-Literal oder AE-Name "EQ" - gleich (Standardwert) "NE" - ungleich "GT" - größer "GE" - größer oder gleich "LT" - kleiner "LE" - kleiner oder gleich "CP" - Mustervergleich (mit Wildcardzeichen) "NP" - Muster ausschließen "BT" - Intervall (auch HIGH angeben) "NB" - außerhalb Intervall (auch HIGH angeben)

Anmerkungen

Das Script-Sprachmittel definiert eine einzelne freie Abgrenzung. Rufen Sie es mehrmals hintereinander auf, um mehrere freie Abgrenzungen festzulegen. Sie werden gesammelt und dem nächsten R3_ACTIVATE_REPORT-Aufruf übergeben.

⚠ Wenn Sie mehrere R3_ACTIVATE_REPORT-Aufrufe im Script des SAP-Jobs verwenden, müssen Sie die freien Abgrenzungen für jeden einzeln setzen. Sie werden nicht für nachfolgende Funktions-Aufrufe übernommen.

Beispiel

```
R3_SET_FREE_SELECTION
TABLENAME='TAB01',FIELDNAME='FIELD01',LOW='17',SIGN='I'
R3_ACTIVATE_REPORT
REPORT='ZSUSER00',COVERPAGE=YES,DESTINATION=LT77,IMMEDIATELY=YES
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_SET_LOG_ATTR

Setzt ein Log-Attribut in der SAP-Monitor-Architektur.

Transaktion: RZ20

Schnittstelle: XMW

Syntax

R3_SET_LOG_ATTR

NODE=...
,MESSAGE=...
[**,VAR1=**...]
[**,VAR2=**...]
[**,VAR3=**...]
[**,VAR4=**...]
[**,COL_METHOD=**...]
[**,AN_METHOD=**...]
[**,AU_METHOD=**...]
[**,VIEW=**...]
[**,VIEW_FRAME=**...]
[**,COLOR=**...]
[**,SEVERITY=**...]
[**,MAX_ALERTS=**...]

Syntaxteil	Beschreibung/Format
NODE=	Name des Knotens Format des Wertes: Script-Literal
MESSAGE=	ID und Nummer, der SAP T100 Nachricht (beide Angaben sind durch ein Leerzeichen getrennt) Format des Wertes: Script-Literal Verwenden Sie "00 398", um nur die variablen Inhalte anzuzeigen.
VAR1=	Nachrichtenvariable 1 Format des Wertes: Script-Literal
VAR2=	Nachrichtenvariable 2 Format des Wertes: Script-Literal
VAR3=	Nachrichtenvariable 3 Format des Wertes: Script-Literal
VAR4=	Nachrichtenvariable 4 Format des Wertes: Script-Literal
COL_ METHOD=	Datensammelmethode des Knotens Format des Wertes: Script-Literal
AN_ METHOD=	Analysemethode des Knotens Format des Wertes: Script-Literal
AU_ METHOD=	Auto-Reaktionsmethode des Knotens Format des Wertes: Script-Literal
VIEW=	Legt fest welche Meldung des Log-Attributes in der aktuellen Statusansicht des Alarm-Monitors angezeigt werden soll. Format des Wertes: Script-Literal Erlaubte Werte: "HIGHALERT", "LAST" (Standardwert) und "WORST_SINCE" "HIGHALERT" - Höchste "LAST" - Letzte "WORST_SINCE" - Schwerwiegendste seit

VIEW_ FRAME=	Dauer in Minuten in der der schwerwiegendste Alarm angezeigt werden soll. Format des Wertes: Zahl Standardwert: "0" Der Wert "0" bewirkt, dass der Standardwert in der Monitor-Architektur verwendet wird. Dieser beträgt "30".
COLOR=	Farbe Format des Wertes: Script-Literal Erlaubte Werte: "AL_VAL_GREEN" (Standardwert), "AL_VAL_YELLOW" und "AL_VAL_RED" "AL_VAL_GREEN" - Grün "AL_VAL_YELLOW" - Gelb "AL_VAL_RED" - Rot
SEVERITY=	Schweregrad Format des Wertes: Zahl Standardwert: "0" Der Wert "0" bewirkt, dass der Standardwert in der Monitor-Architektur verwendet wird. Dieser beträgt "255".
MAX_ ALERTS=	Maximale Alarm-Anzahl, die in dem Knoten gespeichert werden sollen. Format des Wertes: Zahl Standardwert: "0" Diese Beschränkung gilt nur dann, wenn die Alarme in der Monitor-Architektur gespeichert werden.

Anmerkungen

Wenn das Log-Attribut bereits existiert, wird es geändert. Sollte es noch nicht vorhanden sein, legt es das Script-Sprachmittel mit der Meldung an.

Format des Parameters NODE=

Dieser Parameter beschreibt einen kompletten Pfad. Die einzelnen Bestandteile sind durch Schrägstriche "/" getrennt.

Der Pfad beginnt immer mit einem Kontextknoten. Danach können Summenknoten, ein Objektknoten und ein Attributknoten folgen.

Kontext- , Objekt- und Attributknoten dürfen nur einmal im Pfad vorkommen. Summenknoten hingegen mehrmals.

Beispiel 1:

"UC4/TestNode/PerfAttributUC4"

- "UC4" - Kontextname
- "TestNode" - Objektknoten
- "PerfAttributUC4" - Attributknoten

Beispiel 2:

"UC4/Summary1/Summary2/TestNode/PerfAttributUC4"

- "UC4" - Kontextname
- "Summary1" - Summenknoten
- "Summary2" - Summenknoten
- "TestNode" - Objektknoten
- "PerfAttributUC4" - Attributknoten

Beispiel

Nach dem Ausführen des folgenden Beispiels, existiert ein Kontextknoten namens "UC4", ein Objektknoten namens "Test" und ein Log-Attribut mit der Meldung "00 398".

```
R3_SET_LOG_ATTR NODE="UC4/Test/LogAttribute",MESSAGE="00
398",VAR1="variable1",VAR2="variable2",VAR3="variable3",VAR4="variable
4",SEVERITY="12"
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_SET_PERF_ATTR

Setzt ein Performance-Attribut in der SAP-Monitor-Architektur.

Transaktion: RZ20

Schnittstelle: XMW

Syntax

R3_SET_PERF_ATTR

NODE=...
,VALUE=...
[**,UNIT=**...]
[**,COL_METHOD=**...]
[**,AN_METHOD=**...]
[**,AU_METHOD=**...]
[**,ALERT_DIR=**...]
[**,G2Y=**...]
[**,Y2R=**...]
[**,Y2G=**...]
[**,R2Y=**...]

Syntaxteil	Beschreibung/Format
NODE=	Name des Knotens Format des Wertes: Script-Literal
VALUE=	Wert, der gesetzt werden soll Format des Wertes: Zahl
UNIT=	Einheit des Wertes (Abkürzung) Format des Wertes: Script-Literal

COL_ METHOD=	Datensammelmethode des Knotens Format des Wertes: Script-Literal
AN_ METHOD=	Analysemethode des Knotens Format des Wertes: Script-Literal
AU_ METHOD=	Auto-Reaktionsmethode des Knotens Format des Wertes: Script-Literal
ALERT_ DIR=	Legt fest, dass ein Alarm erzeugt wird, wenn der Wert einen Schwellwert über- oder unterschreitet. Erlaubte Werte: "ABOVE" (Standardwert) und "BELOW" "ABOVE" - Alarm für die Überschreitung eines Schwellwertes "BELOW" - Alarm für die Unterschreitung eines Schwellwertes
G2Y=	Schwellwert für die Änderung von Grün zu Gelb Format des Wertes: Zahl Standardwert: "0"
Y2R=	Schwellwert für die Änderung von Gelb zu Rot Format des Wertes: Zahl Standardwert: "0"
Y2G=	Schwellwert für die Änderung von Gelb zu Grün Format des Wertes: Zahl Standardwert: "0"
R2Y=	Schwellwert für die Änderung von Rot zu Gelb Format des Wertes: Zahl Standardwert: "0"

Anmerkungen

Wenn das Performance-Attribut bereits existiert, wird es geändert. Sollte es noch nicht vorhanden sein, legt es das Script-Sprachmittel mit dem Wert an.

Format des Parameters NODE=

Dieser Parameter beschreibt einen kompletten Pfad. Die einzelnen Bestandteile sind durch Schrägstriche "/" getrennt.

Der Pfad beginnt immer mit einem Kontextknoten. Danach können Summenknoten, ein Objektknoten und ein Attributknoten folgen.

Kontext- , Objekt- und Attributknoten dürfen nur einmal im Pfad vorkommen. Summenknoten hingegen mehrmals.

Beispiel 1:

"UC4/TestNode/PerfAttributUC4"

- "UC4" - Kontextname
- "TestNode" - Objektknoten
- "PerfAttributUC4" - Attributknoten

Beispiel 2:

"UC4/Summary1/Summary2/TestNode/PerfAttributUC4"

- "UC4" - Kontextname
- "Summary1" - Summenknoten
- "Summary2" - Summenknoten
- "TestNode" - Objektknoten
- "PerfAttributUC4" - Attributknoten

Beispiel

Nach dem Ausführen des folgenden Beispiels, existiert ein Kontextknoten namens "UC4", ein Objektknoten namens "Test" und ein Performance-Attribut mit dem Wert "150". Die Farbe Grün wird zugewiesen, da der Schwellwert von G2Y nicht überschritten worden ist.

```
R3_SET_PERF_ATTR
NODE="UC4/Test/PerfAttribute",VALUE="150",UNIT="sec",G2Y="160",Y2R="20
0",Y2G="160",R2Y="190"
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_SET_PRINT_DEFAULTS

Setzt Standardwerte für die Druckparameter, die beim Ausführen von Reports verwendet werden.

Transaktion: -

Schnittstelle: AE und Standard

Syntax

R3_SET_PRINT_DEF[AULTS]

[**DESTINATION=**...]
[**,COVERPAGE=**...]
[**,IMMEDIATELY=**...]
[**,RELEASE=**...]
[**,COPIES=**...]
[**,LIST_NAME=**...]
[**,LIST_TEXT=**...]
[**,NEW_LIST_ID=**...]
[**,EXPIRATION=**...]
[**,LINE_COUNT=**...]
[**,LINE_SIZE=**...]
[**,LAYOUT=**...]
[**,SAP_COVER_PAGE=**...]
[**,RECEIVER=**...]
[**,DEPARTMENT=**...]
[**,AUTHORITY=**...]
[**,DATA_SET=**...]
[**,TYPE=**...]
[**,OS_COVER_PAGE=**...]

[**,SPOOL_PRIORITY=**...]
[**,TEXTONLY=**...]
[**,FRAMES=**...]
[,**SUPPRESS_SHADING=**...]
[,**WITH_STRUCTURE=**...]
[,**DEFAULT_SPOOL_SIZE=**...]
[,**PRINTER_MAIL_ADDRESS=**...]
[,**SPOOL_PAGE_FROM=**...]
[,**SPOOL_PAGE_TO=**...]
[**,ARCHIVE_MODE=**...]
[**,ARCHIVE_SAPOBJECT=**...]
[**,ARCHIVE_OBJECT=**...]
[**,ARCHIVE_INFO=**...]
[**,ARCHIVE_TEXT=**...]
[**,MONITOR=**...]

Syntaxteil	Beschreibung/Format
DEST[INATION]=	Ausgabegerät Format: Script-Literal Mit diesem Parameter geben Sie den Namen des Ausgabegerätes an. Dies ist in den meisten Fällen ein Druckername, kann aber auch der Name eines Telefaxgerätes o.ä. sein. ⚠ Beachten Sie, dass die Nutzung der langen SAP-Druckernamen innerhalb von AE aus technischen Gründen, die an der XBP-Schnittstelle von SAP liegen, nicht möglich ist. Geben Sie daher den 4-stelligen technischen Namen an.
Druckparameter	
COVER[PAGE]=	Selektions-Deckblatt Format: Script-Literal Erlaubte Werte: "YES" und "NO" (Standardwert) Dieser Parameter bestimmt, ob am Beginn der Liste ein Deckblatt mit den Reportselektionen ausgegeben werden soll. Wenn ein Deckblatt erzeugt wird, so wird dieses auch in den Jobreport übernommen. Damit ist dokumentiert, mit welchen Parametern diese Durchführung erfolgte.
IMM[EDIATELY]=	Sofort ausgeben Format: Script-Literal Erlaubte Werte: "YES" und "NO" (Standardwert)
REL[EASE]=	Löschen nach Ausgabe Format: Script-Literal Erlaubte Werte: "YES" und"NO" (Standardwert) Mit diesem Parameter wird bestimmt, ob der Spool-Auftrag sofort nach Abschluss der Ausgabe zum Ausgabegerät oder erst nach Ablauf der Spool-Verweildauer gelöscht wird.

COPIES=	Anzahl Ausdrucke Format: Integer Standardwert: "0" Geben Sie hier an, wieviel Exemplare Ihres Dokumentes ausgedruckt werden sollen.
LIST_N[AME]=	Name des Spool-Auftrags Format: Script-Literal Mit diesem Parameter geben Sie den Namen des Spool-Auftrages an. Dieser darf beliebig aus Buchstaben, Ziffern, Sonderzeichen und Leerzeichen bestehen. Der Standardname, der für einen Spool-Auftrag vorgeschlagen wird, besteht aus den 8 Zeichen des Reportnamens, dem Trennzeichen '_' und den ersten 3 Zeichen des Benutzernamens.
LIST_T[EXT]=	Text für Deckblatt Format: Script-Literal Mit diesem Parameter geben Sie den Beschreibungstext zum Spool-Auftrag an. Dieser darf beliebig aus Buchstaben, Ziffern, Sonderzeichen und Leerzeichen bestehen.
NEW_LIST_ID=	Neuer Spool-Auftrag Format: Script-Literal Erlaubte Werte: "YES" (Standardwert) und"NO"
EXPIR[ATION]=	Spool-Verweildauer Format: Integer Standardwert: "0" Dieser Parameter bestimmt, wieviel Tage ein Spool-Auftrag im Spool-System verweilen soll, bevor er gelöscht wird.
LINE_COUNT=	Seitenlänge der Liste Format: Integer Standardwert: "0" Zeilenzahl pro Seite einer Liste. Wenn dieses Feld eine Null enthält oder leer ist, ist die Seitenzahl für die Liste unbegrenzt (beim Drucken nicht erlaubt). Die Länge der Liste wird dann allein durch ihren Inhalt bestimmt. Beim Drucken hängt die maximale Zeilenzahl pro Seite von der gewählten Aufbereitung ab. Wollen Sie die Zeilenzahl ändern, dann müssen Sie eine andere Aufbereitung wählen.
LINE_SIZE=	Zeilenbreite der Liste Format: Integer Standardwert: "0" Dieser Parameter enthält die aktuelle Zeilenbreite der Liste. Beim Drucken hängt die maximale Zeilenbreite von der gewählten Aufbereitung ab. Wollen Sie die Zeilenbreite ändern, so müssen Sie eine andere Aufbereitung wählen.

LAYOUT=	Druckaufbereitung Format: Script-Literal Dieser Parameter bestimmt die Aufbereitung des Spool-Auftrags bei der Ausgabe. Durch die Aufbereitung wird im wesentlichen das Seitenformat definiert, d.h. die maximale Anzahl der Zeilen und Spalten auf einer Druckseite.
SAP_COVER[_PAGE]=	SAP-Deckblatt Format: Script-Literal Erlaubte Werte: "" (Standardwert), "X" und "D" "" - Deckblatt nicht ausgeben "X" - Deckblatt ausgeben "D" - Deckblattausgabe hängt von der Einstellung des jeweiligen Ausgabegerätes (Drucker) ab Dieser Parameter bestimmt, ob bei Ausgabe des Spool-Auftrags ein Deckblatt mit verschiedenen Angaben wie Empfängernamen, Abteilungsname, verwendete Aufbereitung u.ä. ausgegeben werden soll.
RECEIVER=	Empfänger Format: Script-Literal Dieser Parameter enthält den Namen des Empfängers des Spool-Auftrages. Dieser Name wird beim Ausdruck auf dem Deckblatt ausgegeben. Der Standardwert für den Empfängernamen ist der aktuelle Benutzername.
DEPART[MENT]=	Abteilung im Deckblatt Format: Script-Literal Dieser Parameter enthält den Namen der Abteilung zum Spool-Auftrag. Dieser Name wird beim Ausdruck auf dem Deckblatt ausgegeben.
AUTHORITY=	Berechtigung Format: Script-Literal Dieser Parameter enthält die Berechtigung für den Spool-Auftrag (maximal 12 Stellen). Es können sich nur Benutzer den Inhalt des Spool-Auftrags anzeigen lassen, welche die angegebene Berechtigung besitzen.
DATA_SET=	Name des Spool-Datasets Format: Script-Literal
TYPE=	Typ des Spool-Auftrag Format: Script-Literal
OS_COVER[_PAGE]=	Hostspooler-Deckblatt Format: Script-Literal Erlaubte Werte: "" (Standardwert), "X" oder "D" "" - Deckblatt nicht ausgeben "X" - Deckblatt ausgeben "D" - Deckblattausgabe hängt von der Einstellung des jeweiligen Ausgabegerätes (Drucker) ab

SPOOL_PRI[ORITY]=	Priorität eines Spoolauftrages Format: Integer Standardwert: "5"
TEXTO[NLY]=	Nur Text Format: Script-Literal Erlaubte Werte: "YES" und "NO" (Standardwert) Steuert die Ausgabe der Nicht-ASCII-Zeichen einer Druckliste. Voraussetzung für die Nutzung des Parameters (siehe auch SAP Hinweis 777337): • SAP Basis Release 6.20 mit Supportpackage SAPKB62045 • SAP Basis Release 6.40 mit Supportpackage SAPKB64010
FRAMES=	Rahmenzeichen Format: Script-Literal Erlaubte Werte: "YES" (Standardwert) und "NO" Steuert die standardmäßige Rahmenbildung. Für die Nutzung gelten dieselben Voraussetzungen wie für den Parameter TEXTONLY=.
SUPPRESS_ SHADING=	Farben und Schatten nicht drucken. Erlaubte Werte: "YES", "NO" (Standardwert)
WITH_STRUCTURE=	Strukturierte Informationen einbeziehen. Erlaubte Werte: "YES", "NO" (Standardwert)
DEFAULT_SPOOL_ SIZE=	255 Zeichen als maximale Zeilenbreite definieren. Erlaubte Werte: "YES", "NO" (Standardwert) "YES" = nach der 255. Spalte wird ein Zeilenumbruch erzwungen "NO" = keine Beschränkung der Zeilenlänge
PRINTER_MAIL_ ADDRESS=	E-Mail Adresse eines E-Mail Druckers Format: Script-Literal
SPOOL_PAGE_ FROM= **SPOOL_PAGE_TO=**	Nummer der Seite, ab und/oder bis zu welcher gedruckt werden soll. Format: Zahl Standardmässig werden alle Seiten gedruckt.
Archivierungsparameter	Archiv-Parameter bei R3_ACTIVATE_REPORT
ARCHIVE_M[ODE]=	Ablagemodus Format: Script-Literal Erlaubte Werte: "1" (Standardwert), "2" und "3" "1" - Das Dokument wird nur gedruckt. "2" - Das Dokument wird nur im optischen Archiv gespeichert. "3" - Das Dokument wird gedruckt und im optischen Archiv gespeichert.
ARCHIVE_S [APOBJECT]=	Objekttyp des Business Objekts Format: Script-Literal Mittels Objekttypen werden SAP-Objekte klassifiziert.

ARCHIVE_O[BJECT]=	Dokumentart Format: Script-Literal Archivobjekte werden nach Dokumentarten klassifiziert.
ARCHIVE_I[NFO]=	Infofeld Format: Script-Literal Informationskürzel zum Archivierungsauftrag.
ARCHIVE_T[EXT]=	Textinformationsfeld Format: Script-Literal Beschreibungstext zum Archivierungsauftrag. Dieser darf beliebig aus Buchstaben, Ziffern, Sonderzeichen und Leerzeichen bestehen.
Steuernde Parameter	
MON[ITOR]=	Statusüberwachung im Aktivierungslog protokollieren Format: Script-Literal Erlaubte Werte: "YES" und "NO" (Standardwert)

Anmerkungen

Mit dem Script-Sprachmittel können Druckparametern Standardwerte zugewiesen werden. Sie gelten für das gesamte Script oder solange, bis im Script ein neuer Standardwert zugewiesen wird. Die Standardwerte sind nur für den Job gültig, in dem sie gesetzt wurden.

Durch das Setzen von Standardwerten wird die Übersichtlichkeit des AE Scripts erhöht. Gleichzeitig werden Probleme mit der Beschränkung von 1024 Zeichen pro Script-Zeile vermieden, da die eigentliche Verarbeitungsanweisung erst im SAP-Agenten zusammengesetzt wird.

Das Script-Sprachmittel kann ausschließlich für R3_ACTIVATE_REPORT genutzt werden, um den Druckparametern Defaultwerte zuzuweisen.

Beispiel

Im Beispiel werden Standardwerte für das Ausgabegerät und die Anzahl der Ausdrucke gesetzt. Bei der Durchführung des Reports wird die Anzahl der Ausdrucke auf "6" korrigiert. Das gilt aber nur für diesen Report. Der Standardwert ("8") bleibt unverändert.

```
R3_SET_PRINT_DEFAULTS DESTINATION=PR01
R3_SET_PRINT_DEFAULTS COPIES=8
R3_ACTIVATE_REPORT REPORT=RSM04000,COPIES=6
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_SET_SELECT_OPTION

Definiert ein Selektionskriterium.

Transaktion: SA38

Schnittstelle: Standard (XBP 3.0)

Syntax

R3_SET_SELECT_OPTION

,SELNAME=...
,KIND=...
,LOW=...
[**,HIGH=**...]
[**,SIGN=**...]
[**,OPTION=**...]

Syntaxteil	Beschreibung/Format
SELNAME=	Name der Select-Option bzw. des Parameters Format: Script-Literal
KIND=	Art der Selektion Format: Script-Literal Mögliche Werte: "S" und "P" "S" - Selektionskriterium "P" - Parameter
LOW=	Wert des Parameters (LOW) Format: Script-Literal
HIGH=	Wert des Parameters (HIGH) Format: Script-Literal
SIGN=	Kennzeichen (Include/Exclude) Format: Script-Literal Erlaubte Werte: "I" (Standardwert) und "E" "I" = Include "E" = Exclude
OPTION=	Option nach der selektiert werden soll Format: Script-Literal oder AE-Name "EQ" - gleich (Standardwert) "NE" - ungleich "GT" - größer "GE" - größer oder gleich "LT" - kleiner "LE" - kleiner oder gleich "CP" - Mustervergleich (mit Wildcardzeichen) "NP" - Muster ausschließen "BT" - Intervall (auch HIGH angeben) "NB" - außerhalb Intervall (auch HIGH angeben)

Anmerkungen

Das Script-Sprachmittel definiert ein einzelnes Selektionskriterium. Rufen Sie es mehrmals hintereinander auf, um mehrere Selektionskriterien festzulegen. Sie werden gesammelt und dem nächsten R3_ACTIVATE_REPORT- oder R3_CREATE_VARIANT-Aufruf übergeben.

⚠ Wenn Sie mehrere R3_ACTIVATE_REPORT- oder R3_CREATE_VARIANT-Aufrufe im Script des SAP-Jobs verwenden, müssen Sie die Selektionskriterien für jeden einzeln setzen. Sie werden nicht für nachfolgende Funktions-Aufrufe übernommen.

Beispiel

Das Beispiel definiert den Wert "17" für den Parameter "Mindestalter". Das Selektionskriterium wird anschließend für das Anlegen einer Variante verwendet.

```
R3_SET_SELECT_OPTION SELNAME='MIN_ALT',KIND='P',LOW='17',SIGN='I'
R3_CREATE_VARIANT REP=REPORT01,VAR=NEW,TEXT='Neue Variante'
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_SET_STATUS_ATTR

Setzt ein Status-Attribut in der SAP-Monitor-Architektur.

Transaktion: RZ20

Schnittstelle: XMW

Syntax

R3_SET_STATUS_ATTR

NODE=...
,MESSAGE=...
[**,VAR1=**...]
[**,VAR2=**...]
[**,VAR3=**...]
[**,VAR4=**...]
[**,COL_METHOD=**...]
[**,AN_METHOD=**...]
[**,AU_METHOD=**...]
[**,CAUSE_ALERT=**...]
[**,COLOR=**...]

Syntaxteil	Beschreibung/Format
NODE=	Name des Knotens Format des Wertes: Script-Literal
MESSAGE=	ID und Nummer, der SAP T100 Nachricht (beide Angaben sind durch ein Leerzeichen getrennt) Format des Wertes: Script-Literal Verwenden Sie "00 398", um nur die variablen Inhalte anzuzeigen.
VAR1=	Nachrichtenvariable 1 Format des Wertes: Script-Literal

VAR2=	Nachrichtenvariable 2 Format des Wertes: Script-Literal
VAR3=	Nachrichtenvariable 3 Format des Wertes: Script-Literal
VAR4=	Nachrichtenvariable 4 Format des Wertes: Script-Literal
COL_ METHOD=	Datensammelmethode des Knotens Format des Wertes: Script-Literal
AN_ METHOD=	Analysemethode des Knotens Format des Wertes: Script-Literal
AU_ METHOD=	Auto-Reaktionsmethode des Knotens Format des Wertes: Script-Literal
CAUSE_ ALERT=	Legt fest, ob ein Alarm ausgelöst werden soll. Format des Wertes: Script-Literal Erlaubte Werte: "ALWAYS" (Standardwert), "VALUE_CHG", "MSG_CHG" und "NEVER" "ALWAYS" - Immer "VALUE_CHG" - Bei Änderung des Wertes "MSG_CHG" - Bei Änderung der Meldung "NEVER" - Nie
COLOR=	Farbe Format des Wertes: Script-Literal Erlaubte Werte: "AL_VAL_GREEN" (Standardwert), "AL_VAL_YELLOW" und "AL_VAL_RED" "AL_VAL_GREEN" - Grün "AL_VAL_YELLOW" - Gelb "AL_VAL_RED" - Rot

Anmerkungen

Wenn das Status-Attribut bereits existiert, wird es geändert. Sollte es noch nicht vorhanden sein, legt es das Script-Sprachmittel mit der Meldung an.

Format des Parameters NODE=

Dieser Parameter beschreibt einen kompletten Pfad. Die einzelnen Bestandteile sind durch Schrägstriche "/" getrennt.

Der Pfad beginnt immer mit einem Kontextknoten. Danach können Summenknoten, ein Objektknoten und ein Attributknoten folgen.

Kontext- , Objekt- und Attributknoten dürfen nur einmal im Pfad vorkommen. Summenknoten hingegen mehrmals.

Beispiel 1:

"UC4/TestNode/PerfAttributUC4"

- "UC4" - Kontextname
- "TestNode" - Objektknoten

- "PerfAttributUC4" - Attributknoten

Beispiel 2:

"UC4/Summary1/Summary2/TestNode/PerfAttributUC4"

- "UC4" - Kontextname
- "Summary1" - Summenknoten
- "Summary2" - Summenknoten
- "TestNode" - Objektknoten
- "PerfAttributUC4" - Attributknoten

Beispiel

Nach dem Ausführen des folgenden Beispiels, existiert ein Kontextknoten namens "UC4", ein Objektknoten namens "Test" und ein Status-Attribut mit der Meldung "SY 019".

```
R3_SET_STATUS_ATTR NODE="UC4/Test/StatusAttribute",MESSAGE="SY 019"
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_SET_TEXT_ATTR

Setzt ein Text-Attribut in der SAP-Monitor-Architektur.

Transaktion: RZ20

Schnittstelle: XMW

Syntax

R3_SET_TEXT_ATTR

NODE=...
,TEXT=...

Syntaxteil	Beschreibung/Format
NODE=	Name des Knotens Format des Wertes: Script-Literal
TEXT=	Text, der angezeigt werden soll. Format des Wertes: Script-Literal

Anmerkungen

Wenn das Text-Attribut bereits existiert, wird es geändert. Sollte es noch nicht vorhanden sein, legt es das Script-Sprachmittel mit dem Text an.

Format des Parameters NODE=

Dieser Parameter beschreibt einen kompletten Pfad. Die einzelnen Bestandteile sind durch Schrägstriche "/" getrennt.

Der Pfad beginnt immer mit einem Kontextknoten. Danach können Summenknoten, ein Objektknoten und ein Attributknoten folgen.

Kontext- , Objekt- und Attributknoten dürfen nur einmal im Pfad vorkommen. Summenknoten hingegen mehrmals.

Beispiel 1:

"UC4/TestNode/PerfAttributUC4"

- "UC4" - Kontextname
- "TestNode" - Objektknoten
- "PerfAttributUC4" - Attributknoten

Beispiel 2:

"UC4/Summary1/Summary2/TestNode/PerfAttributUC4"

- "UC4" - Kontextname
- "Summary1" - Summenknoten
- "Summary2" - Summenknoten
- "TestNode" - Objektknoten
- "PerfAttributUC4" - Attributknoten

Beispiel

Nach dem Ausführen des folgenden Beispiels, existiert ein Kontextknoten namens "UC4", ein Objektknoten namens "Test" und ein Attribut-Knoten namens "TextAttribute" mit dem Inhalt "Wert".

```
R3_SET_TEXT_ATTR NODE='UC4/Test/TextAttribute',TEXT='Wert'
```

Siehe auch:

Allgemeines zur SAP-JCL

R3_SWITCH_OPMODE

Schaltet die Betriebsart im SAP um.

Transaktion: RZ03

Schnittstelle: AE

Syntax

R3_SWITCH_OPMODE

NAME=...
,SERVER=...

Syntaxteil	Beschreibung/Format

NAME=	Name einer im SAP definierten Betriebsart Format: Script-Literal Kleinbuchstaben werden nicht in Großbuchstaben umgesetzt.
SERVER=	Name eines SAP-Applikationsservers oder "*" für alle SAP-Applikationsserver Format: Script-Literal Der Name eines einzelnen SAP-Applikationsservers muss in der Form *Host_SID_SYSNR* angegeben werden: *Host* - Rechnername *SID* - System-ID von SAP *SYSNR* - Nummer der SAP-Instanz

Beispiel

Die Betriebsart "Nachtbetrieb" wird für alle SAP-Applikationsserver eingeschaltet.

```
R3_SWITCH_OPMODE NAME='Nachtbetrieb',SERVER='*'
```

Siehe auch:

Allgemeines zur SAP-JCL

4.4.2 SAP BCA

BCA_ACTIVATE_PROCESS

Startet und überwacht einen Prozess, der in einem separaten Prozessnetzwerk abläuft.

Transaktion: -

SAP Banking Release: 4.63+

Syntax

BCA_ACTIVATE_PROC[**ESS**]

ID=...
,REPORT=...
[**,VARIANT=**...]
[**,JOBPREFIX=**...]
[**,MAX_APPL_RC=**...]
[**,JOBLOG=**...]
[**,NETWORK_ID=**...]

Syntaxteil	Beschreibung/Format
ID=	Externe Prozess-Identifikation. Format: Script-Literal

REP[ORT]=	Name des ABAP-Programms, welches ausgeführt werden soll. Format: Script-Literal
VAR[IANT]=	Name einer Variante des ABAP-Programms. Format: Script-Literal Standardwert: ""
JOBPRE[FIX]=	Präfix für die Jobnamen des Prozesses, maximal 18 Stellen. Format: Script-Literal Standardwert: "" Die Namen der Jobs, die zum Prozess gehören, werden aus dem angegebenen Präfix, der 10-stelligen laufende Nummer (RunID) des AE Jobs und einer 3-stelligen eindeutigen Nummer gebildet.
MAX_APPL_RC=	Maximaler Anwendungsrückgabwert für normales Ende des Prozesses. Format: Zahl Standardwert: "0" Bis zum angegebenen Anwendungsrückgabewert gilt der Prozess als normal beendet. Ein größerer Anwendungsrückgabewert führt zum Abbruch des AE Jobs.
JOBL[OG]=	Übernahme der Joblogs in den AE-Report. Format: Script-Literal Erlaubte Werte: "YES" und "NO" (Standardwert) "YES" = Joblogs werden in den AE-Report übernommen. "NO" = Joblogs werden nicht in den AE-Report übernommen.
NETWORK_ID=	Name des Prozessnetzwerkes, in dem der Prozess läuft. Format: Script-Literal Standardwert: ""

Anmerkungen

Das Script-Sprachmittel ermöglicht es, einen Prozess zu starten und zu überwachen. Der Prozess wird in einem separaten Prozessnetzwerk gekapselt, so dass nur dieser eine Prozess in diesem Prozessnetzwerk läuft. Mit dem Ende des Prozesses ist somit auch das Prozessnetzwerk beendet.

Die Priorität mit der der Prozess läuft, richtet sich nach dem eingestellten Wert bei der Jobklasse. Diese können Sie in der Registerkarte "SAP" des Jobs auswählen. Standardmäßig wird die Jobklasse "C" verwendet.

Dem Script-Sprachmittel müssen die Parameter ID= und REPORT= mitgegeben werden.

Die weiteren Parameter sind optional. Wenn der auszuführende ABAP keine Variante hat, kann der Parameter VARIANT= leer bleiben oder weggelassen werden. Wird kein Präfix für die Jobnamen des Prozesses angegeben, bildet die Automation Engine automatisch Jobnamen nach dem Muster UC_JOB_<*RunID*(10)>_<*unique_id*(3)>. Die so gebildeten eindeutigen Jobnamen erlauben nach dem Ende des Prozesses den Zugriff auf die Joblogs. Diese können, wenn gewünscht, zur Analyse in den Job-Report übernommen werden. Auch der Name des Prozessnetzwerkes wird von AE automatisch erzeugt, sobald der Parameter NETWORK_ID= nicht genutzt wird. Er wird nach dem Muster UC_NET_<*RunID*(10)> gebildet.

Nach dem normalen Ende des Prozesses, wird das Script weiter prozessiert. Ein Abbruch des Prozesses führt zum sofortigen Abbruch des AE Jobs.

Mit dem Script-Sprachmittel R3_GET_APPLICATIONLOG können Meldungen von Prozessen und Prozessnetzwerken aus dem Anwendungslog gelesen werden. Die Meldungen werden standardmäßig im AE-Report ausgegeben. Alternativ können die Meldungen auch in eine benannte Datei geschrieben werden.

4.4.3 SAP BW

Allgemeines zur SAP BW-JCL

Folgende Übersicht zeigt die JCL Script-Sprachmittel für SAP BW und die zugehörigen Schnittstellen.

Script-Sprachmittel

Script-Sprachmittel	Beschreibung
BW_ACTIVATE_CHAIN	Startet eine Prozesskette, überwacht die Durchführung und speichert deren Protokolle im Aktivierungsreport.
BW_ACTIVATE_INFOPACKAGE	Plant ein InfoPackage oder mehrere für den sofortigen Start ein.
BW_GET_CHAINS	Liest Prozessketten aus dem BW-System. Die verfügbaren Prozessketten werden im Aktivierungsreport oder in einer Datei gespeichert.
BW_GET_INFOPACKAGES	Liest InfoPackages aus dem BW-System. Die verfügbaren InfoPackages werden im Aktivierungsreport oder in einer Datei gespeichert.
BW_RESTART_CHAIN	Setzt eine abgebrochene Prozesskette fort.
BW_SET_CONSTRAINT	Kriterium für den automatischen Wiederanlauf von Child-Prozessen einer Prozessketten setzen.
BW_SET_INFOPACKAGE_SELECTION	Setzt Selektionsparameter, die für das Lesen der InfoPackages aus dem BW-System verwendet werden.

Siehe auch:

Allgemeines zur AE JCL für Anwendungen

BW_ACTIVATE_CHAIN

Startet eine Prozesskette, überwacht die Durchführung und speichert deren Protokolle im Aktivierungsreport.

Transaktion: RSA1

SAP BW-Version: ab 3.0B mit Patch SAPKW30B11

Syntax

BW_ACTIVATE_CHAIN

ID=...
[**,NOFOUND=**...]
[**,ERROR=**...]
[**,PROCESSLOGS=**...]
[**,RESTART=**...]
[**,JOBLOGS=**...]
[**,LONGTEXT=**...]
[**,REPLICATE=**...]
[**,STATUSRETRY=**...]
[**,SCHEDULE=**...]
[**,SYNCHRONOUS=...**]
[**,GET_SPOOL=...**]
[**,COLLECTLOGS=...**]

Syntaxteil	Beschreibung/Format
ID=	Technische Bezeichnung der Prozesskette. Format des Wertes: Script-Literal
NOFOUND=	Behandlung, wenn eine Prozesskette nicht gefunden wurde. Format des Wertes: Script-Literal Erlaubte Werte: "NORMAL" (Standardwert) oder "ABEND" "NORMAL" = Das Script wird fortgesetzt, der AE-Job endet normal. "ABEND" = Das Script wird nicht fortgesetzt, der AE-Job endet abnormal.
ERROR=	Behandlung, wenn eine Prozesskette abnormal endet. Format: Script-Literal Erlaubte Werte: "ABEND" (Standardwert), "IGNORE" oder "SUSPEND" "ABEND" = Das Script wird nicht fortgesetzt, der AE Job endet abnormal. "IGNORE" = Das Script wird fortgesetzt, der AE Job endet normal. "SUSPEND" = Der Job bleibt aktiv, solange bis ein Wiederanlauf der Prozesskette / bzw. deren Child-Prozesse erfolgreich war oder der Job manuell abgebrochen wurde. Dieser Parameter dient dazu um einen Wiederanlauf von Prozessketten / bzw. deren Child-Prozesse durchzuführen, ohne dass der Job endet und somit die Verarbeitung in einem Parent (zum Beispiel einem Workflow) fortgesetzt wird.
PROCESSLOGS=	Ausgabe der Protokolle der einzelnen Prozesse einer Prozesskette. Format: Script-Literal Erlaubte Werte: "YES" (Standardwert), "NO" oder "ERROR" "YES" = Die Protokolle der einzelnen Prozesse werden an das Protokoll der Prozesskette angehängt. "NO" = Es wird nur das Protokoll der Prozesskette ausgegeben. "ERROR" = Die Protokolle der einzelnen Prozesse werden nur im Fehlerfall ausgegeben.

RESTART=	Anzahl der Versuche, eine abgebrochene Prozesskette fortzusetzen. Format: Zahl Standardwert: "0"
JOBLOGS=	Ausgabe der Protokolle des Backgroundjobs zum Prozess. Format: Script-Literal Erlaubte Werte: "YES" (Standardwert), "NO" oder "ERROR" "YES" = Die Protokolle des Backgroundjobs zum Prozess werden ausgegeben. "NO" = Die Protokolle des Backgroundjobs zum Prozess werden nicht ausgegeben. "ERROR" = Die Protokolle des Backgroundjobs zum Prozess werden nur im Fehlerfall ausgegeben.
LONGTEXT=	Ausgabe des Langtextes (Diagnosetext) zu einer Protokollmeldung. Format: Script-Literal Erlaubte Werte: "YES", "NO" oder "ERROR" (Standardwert) "ERROR" = Der Langtext zu einer Protokollmeldung wird nur im Fehlerfall ausgegeben. "YES" = Der Langtext zu einer Protokollmeldung wird ausgegeben. "NO" = Der Langtext zu einer Protokollmeldung wird nicht ausgegeben.
REPLICATE=	Behandlung der Child-Prozesse von Prozessketten. Format des Wertes: Script-Literal Erlaubte Werte: "ALL", "YES" oder "NO" (Standardwert) "ALL" = Alle Children-Prozesse eines Jobs im Automation Engine System replizieren (inklusive der übersprungenen). Sie erscheinen sodann im Aktivitätenfenster des UserInterfaces. Des Weiteren werden Statistiksätze und Reports im Automation Engine System erzeugt. "YES" = Nur abgebrochenen und beendete Children-Prozesse replizieren. Statistiksätze und Reports werden ebenfalls erstellt. "NO" = Es erfolgt keine Nachbildung im Automation Engine System.
STATUSRETRY=	Anzahl der Wiederholungen für die Statusprüfung Format des Wertes: Zahl Standardwert: "0" Prozessketten können aufgrund eines Verhaltens in SAP kurzzeitig einen Beendet-Status aufweisen und danach aber wieder weiterlaufen. Sollte der Agent gerade zu diesem Zeitpunkt den Status ermitteln, meldet er die Prozesskette als beendet. Der Parameter STATUSRETRY= verhindert, dass das Ende der Prozesskette nicht zu früh erkannt wird. Er legt fest wie oft der Endestatus überprüft werden soll. Erst wenn nach n Wiederholungen die Prozesskette noch immer denselben Status hat, geht wird sie der Automation Engine als beendet gemeldet.

SCHEDULE=	Neueinplanung von Prozessketten Format des Wertes: Script-Literal Erlaubte Werte: "YES" (Standardwert) oder "NO" "YES" = Die Prozesskette wird neu eingeplant. "NO" = Es erfolgt keine Neueinplanung der Prozesskette. Der Start der Prozesskette beschleunigt sich dadurch.
SYNCHRONOUS=	Prozesskette synchron ausführen Format des Wertes: Script-Literal Erlaubte Werte: "YES" oder "NO" (Standardwert) "YES" = Synchrone Durchführung. Prozesskette wird dadurch im Dialog Modus anstatt im Hintergrundmodus durchgeführt, wodurch die einzelnen Child-Prozesse seriell abgearbeitet werden. "NO" = Synchrone Durchführung nicht verwenden
GET_SPOOL=	Spoolliste der gestarteten Prozesskette anfordern Format des Wertes: Script-Literal Erlaubte Werte: "YES" oder "NO" (Standardwert) "YES" = Spoolliste anfordern. Diese wird dabei als Text-Datei in das Verzeichnis gespeichert, welches Sie in der INI-Datei des SAP-Agenten mit dem Parameter Download_dir= (Sektion [GLOBAL])festlegen. Der Name dieser Datei setzt sich folgendermaßen zusammen: *<SAP-Job-Count>*_*<Step-Nummer>*_*<Spool-Nummer>*.txt Zusätzlich wird diese Datei beim AE-Job als Job-Output registriert. "NO" = Spooliste wird nicht angefordert
COLLECTLOGS=	Reports von Child-Prozessen der aktivierten Prozesskette in den Job-Report schreiben Werden Child-Prozesse repliziert, so ist deren Protokoll standardmäßig im Report des Jobs UND der Aufgabe des Child-Prozess verfügbar. Diese Informationen sind somit im Job-Report redundant und können daher weggelassen werden. Erlaubte Werte: "YES" (Standardwert) oder "NO" "YES" - Protokoll von Child-Prozessen in den Job-Report schreiben "NO" - Protokoll von Child-Prozessen der Prozesskette nicht in den Job-Report übernehmen ⚠ Beachten Sie, dass die Reports der Child-Prozesse nicht verfügbar sind, wenn die Parameter REPLICATE und COLLECTLOGS beide auf NO gesetzt sind.

Beschreibung

Das Script-Sprachmittel startet eine Prozesskette. Zuerst werden deren Startbedingungen ermittelt. Falls die Prozesskette nicht sofort startet, sondern erst nach einem Ereignis oder zu einer bestimmten Zeit, werden diese Startbedingungen protokolliert.

Nachdem die Prozesskette gestartet wurde, wird sie bis zu ihrem Ende überwacht. Das Protokoll der Prozesskette wird auf jeden Fall im Aktivierungsreport gespeichert. Abhängig von den

Parametern PROCESSLOGS= und JOBLOGS= werden zusätzlich die Protokolle der einzelnen Prozesse bzw. die Protokolle des Backgroundjobs zum Prozess ausgegeben.

Bricht eine Prozesskette ab, bewirkt der Parameter RESTART=, dass wiederholt versucht wird, diese Prozesskette fortzusetzen.

Beispiel

Im Beispiel soll eine Prozesskette aktiviert werden, die vorher vom Benutzer abgefragt wurde. Auch die Werte für die relevanten Parameter des Script-Sprachmittels resultieren aus dieser Abfrage. Dem Script-Sprachmittel werden diese Werte als Script-Variablen übergeben.

```
:BEGINREAD
:    READ  &CHAIN#,"ZTEST,ZSBB1,ZSBB2",'Prozesskette','ZTEST',M
:    READ &ERROR#,"IGNORE,ABEND",'ERROR=','ABEND'
:    READ  &NOFOUND#,"NORMAL,ABEND",'NOFOUND=','NORMAL'
:    READ &LT#,"YES,NO,ERROR",'LONGTEXT=','YES'
:ENDREAD

BW_ACTIVATE_CHAIN
ID='&CHAIN#',ERROR='&ERROR#',NOFOUND='&NOFOUND#',LONGTEXT='&LT#'
```

BW_ACTIVATE_INFOPACKAGES

Plant ein InfoPackage oder mehrere für den sofortigen Start ein.

Transaktion: RSA1

Syntax

BW_ACTIVATE_INFOPACK[AGES]

[**NAME=**...]
[,ERROR=...]
[**,NOFOUND=**...]
[**,MAXRUNTIME=**...]
[**,LOG=**...]

Syntaxteil	Beschreibung/Format
NAME=	Technischer Name eines InfoPackages. Format des Wertes: Script-Literal
ERROR=	Behandlung, wenn ein InfoPackage abnormal endet. Format des Wertes: Script-Literal Erlaubter Wert: "ABEND" (Standardwert) oder "IGNORE" "ABEND" = Das Script wird nicht fortgesetzt, der AE Job endet abnormal. "IGNORE" = Das Script wird fortgesetzt, der AE Job endet normal.

NOFOUND=	Behandlung, wenn InfoPackage nicht gefunden wurde. Format des Wertes: Script-Literal Erlaubte Werte: "NORMAL" (Standardwert) oder "ABEND" "NORMAL" = Das Script wird fortgesetzt, der AE Job endet normal. "ABEND" = Das Script wird nicht fortgesetzt, der AE Job endet abnormal.
MAXRUNTIME=	Maximale Zeit in Sekunden, in der eingeplante InfoPackages überwacht werden. Format des Wertes: Zahl Standardwert: "86400"
LOG=	Legt fest, ob Log-Meldungen von InfoPackages in den Job-Report aufgenommen werden. Erlaubte Werte: "YES" oder "NO" (Standardwert) "YES" - Log-Meldungen werden protokolliert "NO" - Der Job-Report enthält keine Log-Meldungen von InfoPackages.

Anmerkungen

Das Script-Sprachmittel setzt voraus, dass das betreffende InfoPackage nicht eingeplant ist. Der technische Name des InfoPackages kann mit BW_GET_INFOPACKAGES ermittelt werden. Das Script-Sprachmittel ist so lang aktiv, bis der daraus resultierende Ladejob im SAP BW beendet ist. Endet dieser Ladejob nicht normal, führt das zu einem Abbruch des AE Jobs.

Das Script-Sprachmittel kann auch ohne den Parameter **NAME=** verwendet werden. Dadurch ist es möglich, mehrere InfoPackages in einem Schritt für den sofortigen Start einzuplanen. Mit den Parametern **NOFOUND=** und **ERROR=** kann bestimmt werden, was AE tun soll, wenn InfoPackages nicht gefunden werden oder eines der InfoPackes abnormal endet.

Werden InfoPackages in der Administrator Workbench manuell auf den Quality Status "Y" (gelb) gesetzt, würde der AE Job, der sie eingeplant hat, niemals enden. Mit dem Parameter **MAXRUNTIME=** kann festgelegt werden, wie lange eingeplante InfoPackages überwacht werden sollen. Eine Zeitüberschreitung führt, abhängig vom Parameter **ERROR=**, zu einem Abbruch oder zur Fortsetzung des AE Jobs.

⚠ Bei Verwendung der BW 7.x DataSourcen und Transformation Services (ab SAP NetWeaver 04s) muss nach dem Laden der Infopackages ein Datentransferprozess gestartet werden. In diesem Fall empfehlen wir die Definition von geeigneten Prozessketten im BW und die Steuerung dieser mit BW_ACTIVATE_CHAIN.

Beispiele

Das erste Beispiel plant ein InfoPackage für den sofortigen Start ein. Nach maximal einer Stunde endet das Warten auf die Anforderung.

```
BW_ACTIVATE_INFOPACKAGE NAME="ZPAK_
6LX4XTMC3RJ2DF4F09NFIJW98",MAXRUNTIME=3600
```

Im zweiten Beispiel werden alle InfoPackages selektiert, in deren Langtext das Muster "UC4:*Text*" vorkommt. Anschließend werden die gefundenen InfoPackages zum sofortigen Start eingeplant. Jeder Fehler soll zum Abbruch des AE Jobs führen.

```
BW_SET_INFOPACKAGE_SELECTION
SELNAME='TEXTLONG',SIGN='I',OPTION='CP',LOW='UC4:*Text*'
BW_ACTIVATE_INFOPACKAGES NOFOUND='ABEND',ERROR='ABEND'
```

BW_GET_CHAINS

Liest Prozessketten aus dem BW-System. Die verfügbaren Prozessketten werden im Aktivierungsreport oder in einer Datei gespeichert.

Transaktion: RSA1

SAP BW-Version: ab3.0B mit Patch SAPKW30B11

Syntax

BW_GET_CHAIN[**S**]

[**FILE**=...]
[,**ID**=...]
[,**TEXT**=...]

Syntaxteil	Beschreibung/Format
FILE=	Name einer Datei, in welcher die verfügbaren Prozessketten gespeichert werden soll. Format des Wertes: Script-Literal Standardwert: "" Das Ergebnis wird nicht im Report ausgegeben, wenn dieser Parameter verwendet wird.
ID=	Technische Bezeichnung der Prozesskette. Format des Wertes: Script-Literal Standardwert: "" Für die Selektion können Wildcards verwendet werden.
TEXT=	Beschreibung der Prozesskette. Format des Wertes: Script-Literal Standardwert: "" Für die Selektion können Wildcards verwendet werden.
ENCODING=	Encoding für die erzeugte Ausgabedatei (Parameter FILE=). Beispiel: UTF-8 Standardwert: ISO-8859-1 Wird ein nicht unterstütztes bzw. ungültiges Encoding angegeben, so führt dies zum Abbruch des Jobs mit einer entsprechenden Fehlermeldung. In den SAP-Forms ist für dieses Feld eine Eingabehilfe verfügbar, welche alle unterstützten Encodings auflistet.

Anmerkungen

Alle verfügbaren Prozessketten werden in den Aktivierungsreport oder in eine angegebene Datei geschrieben. Die Datei hat einen strukturierten Aufbau. Die erste Spalte (25 Zeichen) enthält den technischen Namen (ID) der Prozesskette und die zweite Spalte (60 Zeichen) deren Langtext. In der dritten Spalte (1 Zeichen) wird die Objektversion ausgegeben. Die Objektversionen haben folgende Bedeutung:

- A = Aktiv
- M = Überarbeitet
- D = Content
- N = Neu

Beispiel

Es werden die verfügbaren Prozessketten abgefragt und in eine Datei geschrieben.

```
BW_GET_CHAINS FILE='..\TEMP\CHAINS.TXT',TEXT='Test*'
```

BW_GET_INFOPACKAGES

Liest InfoPackages aus dem BW-System. Die verfügbaren InfoPackages werden im Aktivierungsreport oder in einer Datei gespeichert.

Transaktion: RSA1

Syntax

BW_GET_INFOPACK[AGES]

[**FILE=**...]
[**,JOB_STATUS=**...]

Syntaxteil	Beschreibung/Format
FILE=	Name einer Datei mit vollständiger Pfadangabe, in der die verfügbaren InfoPackages gespeichert werden.
JOB_STATUS=	Filter auf den Status der Jobs, die über InfoPackages angestartet wurden. Erlaubte Werte: "S", "R", "F", "A" "S" = Job ist eingeplant. "R" = Job läuft gerade. "F" = Job ist beendet. "A" = Job ist abgebrochen.

Anmerkungen

Das Script-Sprachmittel setzt voraus, dass die InfoPackages zuvor mit BW_SET_INFOPACKAGE_SELECTION selektiert wurden. Alle verfügbaren InfoPackages werden in den Aktivierungsreport oder in eine angegebene Datei geschrieben. Die Datei hat einen strukturierten

Aufbau. Die erste Spalte (31 Zeichen) enthält den technischen Namen des InfoPackages, die zweite Spalte (61 Zeichen) dessen Langtext.

Nachdem das Script-Sprachmittel im Script prozessiert wurde, werden alle Selektionstabellen gelöscht.

Beispiel

Im Beispiel werden alle InfoPackages mit Langtext "Land*" außer jenen mit InfoSource-Namen "0COUNTRY" selektiert und in eine Datei geschrieben.

```
BW_SET_INFOPACKAGE_SELECTION SELNAME="TEXTLONG",LOW="Land*",OPTION=CP
BW_SET_INFOPACKAGE_SELECTION
SELNAME="INFOSOURCE",LOW="0COUNTRY",SIGN=E
BW_GET_INFOPACKAGES FILE="c:\temp\infopackages.txt"
```

BW_RESTART_CHAIN

Setzt eine abgebrochene Prozesskette fort.

Transaktion: RSA1

SAP BW-Version: ab 3.0B mit Patch SAPKW30B11

Syntax

BW_RESTART_CHAIN

ID=...
,LOGID=...
[**,NOFOUND=**...]
[**,ERROR=**...]
[**,PROCESSLOGS=**...]
[**,JOBLOGS=**...]
[**,LONGTEXT=**...]
[**,REPLICATE=**...]
[**,GET_SPOOL=...**]
[**,COLLECTLOGS=...**]

Syntaxteil	Beschreibung/Format
ID=	Technische Bezeichnung der Prozesskette. Format des Wertes: Script-Literal
LOGID=	25-stellige Log-ID. Format des Wertes: Script-Literal
NOFOUND=	Behandlung, wenn die Prozesskette nicht gefunden wurde. Format des Wertes: Script-Literal Erlaubte Werte: "NORMAL" (Standardwert) oder "ABEND" "NORMAL" = Das Script wird fortgesetzt, der AE Job endet normal. "ABEND" = Das Script wird nicht fortgesetzt, der AE Job endet abnormal.

ERROR=	Behandlung, wenn die Prozesskette abnormal endet. Format des Wertes: Script-Literal Erlaubte Werte: "ABEND" (Standardwert), "IGNORE" oder "SUSPEND" "ABEND" = Das Script wird nicht fortgesetzt, der AE Job endet abnormal. "IGNORE" = Das Script wird fortgesetzt, der AE Job endet normal. "SUSPEND" = Der Job bleibt aktiv, solange bis ein Wiederanlauf der Prozesskette / bzw. deren Child-Prozesse erfolgreich war oder der Job manuell abgebrochen wurde. Dieser Parameter dient dazu um einen Wiederanlauf von Prozessketten / bzw. deren Child-Prozesse durchzuführen, ohne dass der Job endet und somit die Verarbeitung in einem Parent (zum Beispiel einem Workflow) fortgesetzt wird.
PROCESSLOGS=	Ausgabe der Protokolle der einzelnen Prozesse einer Prozesskette. Format des Wertes: Script-Literal Erlaubte Werte: "YES" (Standardwert), "NO" oder "ERROR" "YES" = Die Protokolle der einzelnen Prozesse werden an das Protokoll der Prozesskette angehängt. "NO" = Es wird nur das Protokoll der Prozesskette ausgegeben. "ERROR" = Die Protokolle der einzelnen Prozesse werden nur im Fehlerfall ausgegeben.
JOBLOGS=	Ausgabe der Protokolle des Backgroundjobs zum Prozess. Format des Wertes: Script-Literal Erlaubte Werte: "YES" (Standardwert), "NO" oder "ERROR" "YES" = Die Protokolle des Backgroundjobs zum Prozess werden ausgegeben. "NO" = Die Protokolle des Backgroundjobs zum Prozess werden nicht ausgegeben. "ERROR" = Die Protokolle des Backgroundjobs zum Prozess werden nur im Fehlerfall ausgegeben.
LONGTEXT=	Ausgabe des Langtextes (Diagnosetext) zu einer Protokollmeldung. Format des Wertes: Script-Literal Erlaubte Werte: "YES", "NO" oder "ERROR" (Standardwert) "ERROR" = Der Langtext zu einer Protokollmeldung wird nur im Fehlerfall ausgegeben. "YES" = Der Langtext zu einer Protokollmeldung wird ausgegeben. "NO" = Der Langtext zu einer Protokollmeldung wird nicht ausgegeben.

REPLICATE=	Behandlung der Child-Prozesse von Prozessketten. Format des Wertes: Script-Literal Erlaubte Werte: "ALL", "YES" oder "NO" (Standardwert) "ALL" = Alle Children-Prozesse eines Jobs im Automation Engine System replizieren (inklusive der übersprungenen). Sie erscheinen sodann im Aktivitätenfenster des UserInterfaces. Des Weiteren werden Statistiksätze und Reports im Automation Engine System erzeugt. "YES" = Nur abgebrochenen und beendete Children-Prozesse replizieren. Statistiksätze und Reports werden ebenfalls erstellt. "NO" = Es erfolgt keine Nachbildung im Automation Engine System.
GET_SPOOL=	Spoolliste der gestarteten Prozesskette anfordern Format des Wertes: Script-Literal Erlaubte Werte: "YES" oder "NO" (Standardwert) "YES" = Spoolliste anfordern. Diese wird dabei als Text-Datei in das Verzeichnis gespeichert, welches Sie in der INI-Datei des SAP-Agenten mit dem Parameter Download_dir= (Sektion [GLOBAL])festlegen. Der Name dieser Datei setzt sich folgendermaßen zusammen: *<SAP-Job-Count>_<Step-Nummer>_<Spool-Nummer>*.txt Zusätzlich wird diese Datei beim AE-Job als Job-Output registriert. "NO" = Spooliste wird nicht angefordert
COLLECTLOGS=	Reports von Child-Prozessen der aktivierten Prozesskette in den Job-Report schreiben Werden Child-Prozesse repliziert, so ist deren Protokoll standardmäßig im Report des Jobs UND der Aufgabe des Child-Prozess verfügbar. Diese Informationen sind somit im Job-Report redundant und können daher weggelassen werden. Erlaubte Werte: "YES" (Standardwert) oder "NO" "YES" - Protokoll von Child-Prozessen in den Job-Report schreiben "NO" - Protokoll von Child-Prozessen der Prozesskette nicht in den Job-Report übernehmen ⚠ Beachten Sie, dass die Reports der Child-Prozesse nicht verfügbar sind, wenn die Parameter REPLICATE und COLLECTLOGS beide auf NO gesetzt sind.

Beschreibung

Das Script-Sprachmittel setzt eine abgebrochene Prozesskette fort. Mit dem Parameter LOGID= wird dabei eine bestimmte Durchführung dieser Prozesskette eindeutig identifiziert. Die Log-ID kann mit PREP_PROCESS_REPORT aus dem Report eines AE Jobs ermittelt werden.

Nachdem die Prozesskette fortgesetzt wurde, wird sie bis zu ihrem Ende überwacht. Das Protokoll der Prozesskette wird auf jeden Fall im Aktivierungsreport gespeichert. Abhängig von den Parametern PROCESSLOGS= und JOBLOGS= werden zusätzlich die Protokolle der einzelnen Prozesse bzw. die Protokolle des Backgroundjobs zum Prozess ausgegeben.

Beispiel

Im Beispiel wird im Post-Script eines Jobs anhand der Fehlernummer ermittelt, ob die Prozesskette "ZSBB1" abgebrochen ist. In diesem Fall wird die 25-stellige Log-ID in einer Variablen gespeichert.

```
:SET  &HND# = PREP_PROCESS_REPORT
(,,PLOG,'*U2004111*','COL=DELIMITER',"DELIMITER=@'@")
:SET  &LOGID# = ''

:PROCESS  &HND#
:   SET  &LOGID# = GET_PROCESS_LINE(&HND#,4)
:ENDPROCESS

:IF  &LOGID# <> ''
:   PUT_VAR VARA.CHAINS,'ZSBB1',&LOGID#
:ENDIF
```

Ein weiterer Job liest die Log-ID aus der Variablen und startet die Prozesskette erneut, um die Verarbeitung fortzusetzen.

```
:SET  &LOGID = GET_VAR(VARA.CHAINS,ZSBB1)
BW_RESTART_CHAIN
ID='ZSBB1',LOGID='&LOGID#',ERROR='ABEND',NOFOUND='NORMAL'
```

BW_SET_CONSTRAINT

Kriterium für den automatischen Wiederanlauf von Child-Prozessen einer Prozessketten setzen.

Syntax

BW_SET_CONSTRAINT

[TYPE=...]
[,NAME=...]
,FIELD=...
[,OPERATOR=...]
,VALUE=...
,ACTION=...
[,COUNT=...]
[,DELAY=...]

Syntaxteil	Beschreibung/Format
TYPE=	Typ des Child-Prozesses Format des Wertes: Script-Literal Das Wildcardzeichen * kann ebenfalls verwendet werden (Platzhalter für beliebig viele Zeichen). Beispiel: ABAP
NAME=	Name des Child-Prozesses Format des Wertes: Script-Literal Die Wildcardzeichen * und ? (Platzhalter für ein beliebiges Zeichen) stehen ebenfalls zur Verfügung.

FIELD=	Stelle, die nach dem Wert (Parameter VALUE) durchsucht werden soll Format des Wertes: Script-Literal Erlaubte Werte: "STATUS", "JOBLOG" oder "PROCESSLOG" "STATUS" = Status des Child-Prozesses "JOBLOG" = Jobprotokoll. Protokolle des Backgroundjobs "PROCESSLOG" = Prozessprotokoll. Protokolle der Child-Prozesse ⚠ Beachten Sie, dass das Prozessprotokoll und Jobprotokoll nur durchsucht werden können, wenn dies für die aktivierte bzw. neu gestartete Prozesskette aktiviert wurde (Sprachmittel BW_ACTIVATE_CHAIN / BW_RESTART_CHAIN, Parameter JOBLOGS / PROCESSLOGS).
OPERATOR=	Operator für die Bedingung Format des Wertes: Script-Literal Erlaubte Werte: "EQ" (Standardwert), "NE", "CP" oder "NP" "EQ" = Entspricht (nur bei FIELD=STATUS) "NE" = Entspricht nicht (nur bei FIELD=STATUS) "CP" = Enthält Muster "NP" = Enthält Muster nicht
VALUE=	Wert, nach dem gesucht werden soll Format des Wertes: Script-Literal Wird auf den Status der Child-Prozesse geprüft (FIELD=STATUS) so sind folgende Werte möglich (entsprechen der Definition der RSPC_STATE Domain im SAP-System): "X" = Abgebrochen "R" = Fehlerhaft beendet "G" = Erfolgreich beendet "F" = Abgeschlossen "A" = Aktiv "P" = Geplant "S" = Beim Wiederaufsetzen übergangen "Q" = Freigegeben "Y" = Bereit "" = Undefiniert
ACTION=	Aktion, die für den Child-Prozess durchgeführt werden soll, der die Bedingung erfüllt. Format des Wertes: Script-Literal Erlaubte Werte: "RESTART" oder "ABEND" "RESTART" = Wiederanlauf des entsprechenden Child-Prozesses durchführen "ABEND" = Entsprechenden Child-Prozess abbrechen
COUNT=	Anzahl der Wiederanläufe, die für den zutreffenden Child-Prozess durchgeführt werden sollen (nur relevant bei ACTION=RESTART) Format des Wertes: Zahl Standardwert: 1

DELAY=	Zeit in Minuten, die zwischen mehreren Wiederanläufen gewartet werden soll. (nur relevant bei ACTION=RESTART) Format des Wertes: Zahl Standardwert: 1

Beschreibung

Dieses Sprachmittel dient dazu, um einen automatischen Wiederanlauf für Child-Prozesse von Prozessketten durchzuführen.

Mit der Funktion legen Sie ein Kriterium fest Dieses setzt sich aus einer zu prüfenden Bedingung (zum Beispiel: Prüfung auf einen bestimmten Status) und einer durchzuführenden Aktion (Beispiel: Aufgabe abbrechen) zusammen. Für alle Child-Prozesse, die auf diese Bedingung zutreffen, wird die definierte Aktion durchgeführt.

Beispiel

Folgendes Beispiel zeigt die Definition eines Wiederanlauf-Kriteriums, wodurch alle abgebrochenen Child-Prozesse vom Typ "ABAP" deren Name mit "RSM" beginnt, bis zu 3 mal im Intervall von 5 Minuten automatisch neu gestartet werden.

```
BW_SET_CONSTRAINT
NAME="RSM*",VALUE="X",OPERATOR="EQ",ACTION="RESTART",TYPE="ABAP",DELAY
="5",COUNT="3"
```

Durch folgende Definition, werden alle ABAP Child-Prozesse, deren Name mit "RSM" beginnt, abgebrochen, wenn im Protokoll des Background-Jobs der Begriff "error" vorkommt.

```
BW_SET_CONSTRAINT
NAME="RSM*",VALUE="error",OPERATOR="CP",ACTION="ABEND",TYPE="ABAP",DEL
AY="5",COUNT="3",SOURCE="JOBLOG"
```

BW_SET_INFOPACKAGE_SELECTION

Setzt Selektionsparameter, die für das Lesen der InfoPackages aus dem BW-System verwendet werden.

Transaktion: RSA1

Syntax

BW_SET_INFOPACKAGE_SEL[ECTION]

SELNAME=...
,LOW=...
[**,HIGH**=...]
[**,SIGN**=...]
[**,OPTION**=...]
[**,MODE**=...]

Syntaxteil	Beschreibung/Format

SELNAME=	Name einer Selektionstabelle für InfoPackages. Erlaubte Werte: "TEXTLONG", "INFOSOURCE", "SOURCESYSTEM", "DATASOURCE" "TEXTLONG" = Selektion nach Langtexten "INFOSOURCE" = Selektion nach InfoSources "SOURCESYSTEM" = Selektion nach Quellsystemen "DATASOURCE" = Selektion nach Datenquellen
LOW=	Untergrenze zur gewählten Selektionstabelle.
HIGH=	Obergrenze zur gewählten Selektionstabelle.
SIGN=	Kennzeichen für die Selektion. Erlaubte Werte: "I" (Standardwert), "E" "I" = Gefundene InfoPackages werden eingeschlossen. "E" = Gefundene InfoPackages werden ausgeschlossen.
OPTION=	Selektionsoption. Erlaubte Werte: "EQ" (Standardwert), "NE", "GT", "GE", "LT", "LE", "CP", "NP", "BT", "NB" "EQ" = gleich "NE" = ungleich "GT" = größer "GE" = größer oder gleich "LT" = kleiner "LE" = kleiner oder gleich "CP"= Mustervergleich (mit Wildcards) "NP" = Muster ausschließen "BT" = Intervall (auch HIGH angeben) "NB" =- außerhalb Intervall (auch HIGH angeben)
MODE=	Verarbeitungsmodus für die Selektion. Erlaubte Werte: "R" (Standardwert), "A" "R" - Selektion wird in der Selektionstabelle ersetzt. "A" - Selektion wird der Selektionstabelle hinzugefügt.

Anmerkungen

Dieses Script-Sprachmittel führt nicht zu einem SAP-Funktionsaufruf. Es bereitet das nachfolgende Lesen der InfoPackages aus dem BW-System vor. InfoPackages können auf vier verschiedene Arten selektiert werden: Langtext, InfoSource, Quellsystem und Datenquelle.

Bei jedem AE Job für das Zielsystem SAP BW sind die Selektionstabellen anfangs leer. Die mehrmalige Verwendung des Script-Sprachmittels bewirkt, dass die Selektionstabellen nach und nach gefüllt werden.

Unter- und Obergrenze beziehen sich auf die gewählte Selektionstabelle. Wurde beispielsweise "TXTLONG" angegeben, selektieren diese Parameter den Langtext der verfügbaren InfoPackages. Für Unter- und Obergrenze können die Wildcardzeichen "*" und "?" verwendet werden. "*" steht dabei für beliebige, "?" für genau ein Zeichen. Eine Obergrenze ist nur dann sinnvoll, wenn "BT" oder "NB" als Selektionsoption angegeben werden.

Obergrenze, Kennzeichen für die Selektion, Selektionsoption und Verarbeitungsmodus sind optionale Parameter des Script-Sprachmittels.

Das Script-Sprachmittel SET_INFOPACKAGE_SELECTION wurde mit Version 3.0 in BW_SET_INFOPACKAGE_SELECTION umbenannt. Es kann aber auch weiterhin mit seinem bisherigen Namen verwendet werden.

Beispiel

Im Beispiel werden alle InfoPackages mit Langtext "Land*" außer jenen mit InfoSource-Namen "0COUNTRY" selektiert.

```
BW_SET_INFOPACKAGE_SELECTION
SELNAME="TEXTLONG",OPTION="CP",LOW="Land*"
BW_SET_INFOPACKAGE_SELECTION
SELNAME="INFOSOURCE",LOW="0COUNTRY",SIGN=E
```

4.4.4 SAP XI

XI_GET_CHANNEL

Listet Kommunikationskanäle auf.

Syntax

XI_GET_CHANNEL

[**CHANNEL=**...]
[**,SERVICE=**...]
[**,PARTY=**...]
[**,STATE=**...]
[**,ACTSTATE=**...]
[**,NOFOUND=**...]

Syntaxteil	Beschreibung/Format
CHANNEL=	Name des Kommunikationskanals Format des Wertes: Script-Literal Standardwert: "*" Es kann das Wildcardzeichen "*" im Channelnamen verwendet werden.
SERVICE=	Name des Services Format des Wertes: Script-Literal Standardwert: "*" Es kann das Wildcardzeichen "*" im Servicenamen verwendet werden.
PARTY=	Name des Partners Format des Wertes: Script-Literal Standardwert: "*" Es kann das Wildcardzeichen "*" im Partnernamen verwendet werden.

STATE=	Zustand des Kommunikationskanals Format des Wertes: Script-Literal Erlaubte Werte: "*" (Standardwert) "ERROR" "OK" "INACTIVE" "UNKNOWN" "UNREGISTERED"
ACTSTATE=	Status des Kommunikationskanals Erlaubte Werte: "*" (Standardwert), "STARTED" und "STOPPED"
NOFOUND=	Behandlung, wenn keine Kommunikationskanäle, die den Filterkriterien entsprechen, gefunden werden. Format des Wertes: Script-Literal Erlaubte Werte: "NORMAL" (Standardwert) oder "ABEND" "NORMAL" = Der AE-Job läuft weiter. "ABEND" = Der AE-Job endet abnormal.

Anmerkungen

Die Funktion listet die Kommunikationskanäle, die den gesetzten Filterkriterien entsprechen, im Report des Jobs auf. Standardmäßig schränken die Filterparameter die Selektion der Kommunikationskanäle nicht ein.

Der Parameter NOFOUND= erlaubt Ihnen den Job abzubrechen, wenn die Filterbedingungen auf keine der Kommunikationskanäle zutreffen.

Das Ergebnis wird als XML-Dokument im Jobreport hinterlegt. Es kann mit den Script-Sprachmitteln für XMLs ausgelesen werden.

Beispiel für die XML-Ausgabe in den Report:

```
<Report>
<Channels>
<Channel>
<Party/>
<Service>FileListReceiver</Service>
<ChannelName>FileListReceiverChannel</ChannelName>
<ChannelID>85e07c39f9b831d1817f3c4bec0af8ff</ChannelID>
<ActivationState>STARTED</ActivationState>
<ChannelState>OK</ChannelState>
</Channel>
<Channel>
<Party/>
<Service>X64_107</Service>
<ChannelName>GeneratedReceiverChannel_RFC</ChannelName>
<ChannelID>19729e747d023ff7a27d32d3f566ed79</ChannelID>
<ActivationState>STARTED</ActivationState>
<ChannelState>OK</ChannelState>
</Channel>
<Channel>
<Party/>
<Service>SenderListe</Service>
```

```
<ChannelName>File_Sender_Liste</ChannelName>
<ChannelID>8bdda1b7041b37efa313219ae7029906</ChannelID>
<ActivationState>STARTED</ActivationState>
<ChannelState>OK</ChannelState>
</Channel>
</Channels>
</Report>
```

Beispiel

Das Beispiel listet alle inaktiven Kommunikationskanäle auf.

```
XI_GET_CHANNEL STATE='INACTIVE'
```

Hier werden alle Kommunikationskanäle im Report aufgelistet deren Name mit "File" beginnt und die gestartet sind.

```
XI_GET_CHANNEL CHANNEL='File*',ACTSTATE='STARTED'
```

Siehe auch:

XI_SET_CHANNEL

XI_SET_CHANNEL

Startet und stoppt Kommunikationskanäle.

Syntax

XI_SET_CHANNEL

ACTION=...
[**CHANNEL=**...]
[**,SERVICE=**...]
[**,PARTY=**...]
[**,NOFOUND=**...]
[**,ERROR=**...]

Syntaxteil	Beschreibung/Format
ACTION=	Aktion auf den Kommunikationskanal Erlaubte Werte: "START" und "STOP"
CHANNEL=	Name des Kommunikationskanals Format des Wertes: Script-Literal Standardwert: "*" Es kann das Wildcardzeichen "*" im Channelnamen verwendet werden.

SERVICE=	Name des Services Format des Wertes: Script-Literal Standardwert: "*" Es kann das Wildcardzeichen "*" im Servicenamen verwendet werden.
PARTY=	Name des Partners Format des Wertes: Script-Literal Standardwert: "*" Es kann das Wildcardzeichen "*" im Partnernamen verwendet werden.
NOFOUND=	Behandlung, wenn keine Kommunikationskanäle, die den Filterkriterien entsprechen, gefunden werden. Format des Wertes: Script-Literal Erlaubte Werte: "NORMAL" oder "ABEND" (Standardwert) "NORMAL" = Der AE-Job läuft weiter. "ABEND" = der AE-Job endet abnormal.
ERROR=	Behandlung, wenn einer die Aktion für einen der Kommunikationskanäle nicht durchgeführt werden kann. Erlaubte Werte: "IGNORE" und "ABEND" (Standardwert) "IGNORE" = Der AE-Job läuft weiter. "ABEND" = der AE-Job endet abnormal.

Anmerkungen

Die Funktion kann einen oder mehrere Kommunikationskanäle starten bzw. stoppen. Geben Sie bei den Parametern CHANNEL=, SERVICE= und PARTY= entweder gezielt einen Kommunikationskanal oder Filter für mehrere Kommunikationskanäle an.

Mit dem Parameter ERROR= können Sie den Job abbrechen, wenn auf einen der selektierten Kommunikationskanäle der Start bzw. Stop nicht durchführbar ist. Aufgrund des Schnittstellenverhaltens wendet der Agent die ACTION= trotzdem bei allen weiteren Kommunikationskanälen an.

Die Information zum Kommunikationskanal, der nicht gesteuert werden konnte, wird als XML-Dokument im Jobreport hinterlegt. Es kann mit den Script-Sprachmitteln für XMLs ausgelesen werden.

Beispiel für die XML-Ausgabe in den Report:

```
<Report>
<Channels>
<Channel>
<Party/>
<Service>SenderListe</Service>
<ChannelName>SenderChannel</ChannelName>
<ChannelID> f2d7791276e8388b995afd2d7a22e1b0</ChannelID>
<ActivationState>STARTED</ActivationState>
<ChannelState>OK</ChannelState>
<ErrorInformation>
com.sap.aii.af.service.administration.impl.WrongAutomationModeExceptio
n:
The channel "/SenderListe/SenderChannel (GUID
```

```
f2d7791276e8388b995afd2d7a22e1b0)"
is configured to use an automation mode not compatible with the type
of the
current principal (WSUSER). The channel was not started. Change the
channels
automation mode and repeat the administrative action.
</ErrorInformation>
</Channel>
</Channels>
</Report>
```

Beispiel

Das Beispiel startet einen Kommunikationskanal, der über den Namen und das Service gefiltert wird.

```
XI_SET_CHANNEL ACTION='START',CHANNEL=' File_Sender_Liste',SERVICE='
SenderListe',PARTY='*'
```

Hier wird ein Kommunikationskanal gestoppt.

```
XI_SET_CHANNEL ACTION='STOP',CHANNEL='SenderChannel',SERVICE='
SenderListe',PARTY='*'
```

Siehe auch:

XI_GET_CHANNEL

4.5 Siebel

4.5.1 SI_START_TASK

Führt Kommandos in Siebel aus.

Syntax

SI_START_TASK CMD=*Siebel-Kommando*

Syntaxteil	Beschreibung/Format
CMD=	Siebel-Kommando Format des Wertes: Script-Literal

Anmerkungen

Um Kommandos in Siebel ausführen zu lassen, sind diese in der Script-Registerkarte unter Verwendung des Script-Sprachmittels SI_START_TASK einzutragen. Das Siebel-Kommando wird

in weiterer Folge an den Siebel Server Manager (über dessen Parameter /c) weitergegeben und auf diesem Wege in Siebel durchgeführt.

Jede Scriptzeile mit "SI_START_TASK CMD=" ist hierbei eine eigene Aufgabe in Siebel. Beachten Sie, dass das Siebel-Kommando unbedingt mit "start task" beginnen muss, damit eine Überwachung der Aufgabe (z.B.: Abbrechen, Fortsetzen) möglich ist. Verwenden Sie "run task" so wird der Siebel-Job abgebrochen und eine Fehlermeldung ausgegeben.

Beispiel:

```
SI_START_TASK CMD="start task for component EIM server ESERV01 with
Config=apd_contacts_ok.ifb, LovLang=DEU, ErrorFlags=1, SQLFlags=8,
TraceFlags=1"
```

Siehe auch:

Allgemeines zur AE JCL für Anwendungen

5 AE JCL für JMX

5.1 Allgemeines zur JMX-JCL

Folgendes Dokument bietet eine Übersicht aller JCL-Scriptsprachmittel für JMX und den unterstützten Datentypen.

Die folgenden Datentypen werden bei der Übergabe von Werten bei den Script-Sprachmitteln unterstützt:

- java.lang.String
- java.lang.Integer
- java.lang.Byte
- java.lang.Float
- java.lang.Double
- java.lang.Short
- java.lang.Long
- java.lang.BigInteger
- java.lang.Boolean
- java.lang.BigDecimal
- java.lang.Character
- javax.management.openmbean.CompositeData

Für die Rückgabewerte der Script-Sprachmittel können Sie Arrays oder die beiden nachstehenden komlexen Datentypen verwenden:

- javax.management.openmbean.CompositeData
- javax.management.openmbean.TabularData

Die Script-Sprachmittel schreiben ihre Ergebnisse in den Report. Von dort sind sie mittels PREP_PROCESS_REPORT auslesbar.

Script-Sprachmittel

Script-Sprachmittel	Beschreibung
JMX_COMPOSITE_ADD	Fasst Werte für einen Parameter zusammen.
JMX_CREATE_MBEAN	Registriert ein MBean.
JMX_GET_AGENT	Ermittelt die Agent ID des MBean Servers.
JMX_GET_ATTRIBUTE	Liefert den Wert eines oder aller Attribute eines MBeans.
JMX_GET_INFO	Gibt Informationen über ein MBean aus.
JMX_INVOKE	Ruft eine Funktion eines MBeans auf.
JMX_QUERY_NAMES	Listet registrierte MBeans auf.
JMX_SET_ATTRIBUTE	Setzt ein Attribut eines MBeans.

JMX_WAIT_FOR_NOTIFICATION	Wartet auf eine Benachrichtigung des MBeans.
JMX_UNREGISTER_MBEAN	Entfernt ein bereits registriertes MBean.

Siehe auch:

Registerkarte **Form** (JMX)

5.2 JMX_COMPOSITE_ADD

Fasst Werte für einen Parameter zusammen.

Syntax

JMX_COMPOSITE_ADD NAME=..., KEY=..., VALUE=...

Syntaxteil	Beschreibung/Format
NAME=	Name für den Parameter
KEY=	Schlüsselbegriff Welche Bezeichnung Sie hier wählen können, hängt vom MBean ab, dem Sie den Parameter übergeben möchten.
VALUE=	Wert, der unter dem Schlüsselbegriff gespeichert werden soll

Anmerkungen

Viele MBeans verlangen anstatt eines einzelnen eine ganze Liste von Werten für ihre Parameter. JMX_COMPOSITE_ADD speichert mehrere Werte unter einem Parameternamen ab. Sie können die Werte nur innerhalb des Jobs verwenden.

Die Parameterwerte existieren bis zum Ende des Jobs und können für jedes beliebige MBean verwendet werden.

In der folgende Zeile definiert einen einzelnen Wert.

```
JMX_COMPOSITE_ADD NAME="parameter",KEY="CUSTOMERSTATE",VALUE="Partner"
```

Besteht die Angabe für einen Schlüsselbegriff aus mehr als einem Wert, so sind die einzelnen Teilwerte mit ";" zu trennen.

```
JMX_COMPOSITE_ADD NAME="parameter",KEY="COUNTRY",VALUE="USA;India"
```

Der ";" kann auch innerhalb eines Wertes vorkommen. Fügen Sie in diesem Fall ein zusätzliches "\" direkt vor dem Strichpunkt ein. Dadurch wird er nicht als Trennzeichen interpretiert.

Bereiche mit Minimal- und Maximalwert werden ebenfalls getrennt durch ";" angegeben.

```
JMX_COMPOSITE_ADD NAME="parameter",KEY="ID",VALUE="33333;44444"
```

Das Format für ein Datum ist JJJJMMTT, für eine Uhrzeit HHMMSS und für eine Datums plus Uhrzeitangabe JJJJMMTTHHMMSS.

```
JMX_COMPOSITE_ADD NAME="parameter",KEY="TIME",VALUE="121516"
```

```
JMX_COMPOSITE_ADD
NAME="parameter",KEY="DATEOPENED",VALUE="19990616120000"
```

Bei Werten vom Typ Boolean wird entweder "true" oder "false" verwendet.

```
JMX_COMPOSITE_ADD NAME="parameter",KEY="REQUEST",VALUE="true"
```

Für Werte vom Typ Währung geben Sie nur den Zahlenwert an.

```
JMX_COMPOSITE_ADD NAME="parameter",KEY="CURRENCY",VALUE="1000.00"
```

⚠ Tragen Sie den mehrzeiligen Wert direkt in der Registerkarte "Script" ein. Die Zeile sollte anschließend nicht mehr in der Registerkarte **Form** editiert werden, da sie keine mehrzeiligen Texte unterstützt und daher nur die erste Zeile übernehmen würde.

⚠ Vermeiden Sie Sonderzeichen und Umlaute in Parameternamen und Werten. Der JMX-Agent sendet alle Web Service Requests in UTF-8. Ob die Konvertierung der JCL erfolgreich ist, hängt vom Encoding ab, das der Administrator in der Variablen UC_SYSTEM_SETTINGS im Key XML_ENCODING eingestellt hat. Der Agent geht davon aus, dass "ISO-8859-15" eingetragen ist.

Beispiel

Im Beispiel wird der Parameter "report" des MBeans für Crystal Reports befüllt. Es erhält Werte für das Dateiformat sowie den Absender und den Betreff für ein E-Mail.

```
JMX_COMPOSITE_ADD NAME="report",KEY="FORMAT",VALUE="EXCEL"
JMX_COMPOSITE_ADD NAME="report",KEY="MAIL_SUBJECT",VALUE="Kunden in
Wien"
JMX_COMPOSITE_ADD NAME="report",KEY="MAIL_FROM",VALUE="meier@uc4.com"
```

Siehe auch:

AE JCL für JMX

5.3 JMX_CREATE_MBEAN

Registriert ein MBean.

Syntax

JMX_CREATE_MBEAN CLASSNAME=... [, NAME=...] [, LOADERNAME=...] [EXISTS=...]

Syntaxteil	Beschreibung/Format
CLASSNAME=	Name der Klasse des MBeans.
NAME=	Name mit dem das MBean registriert werden soll. Sie müssen den Parameter nicht angeben, wenn das MBean das MBeanRegistration Interface implementiert.
LOADERNAME=	Name der Ladeklasse. Standardmäßig wird das Default Loader Repository zum Laden des MBeans verwendet.

EXISTS=	Behandlung des AE-Jobs, wenn das MBean schon registriert ist. Mögliche Werte: "IGNORE" und "ABORT" (Standardwert) "IGNORE" - Das Script des JMX-Jobs wird fortgesetzt. "ABORT" - Der JMX-Job bricht ab.

Anmerkungen

Das Script-Sprachmittel erzeugt eine Instanz der angegebenen Klasse und registriert sie im MBean Server mit dem Objektnamen. Die Parameter LOADERNAME= kann einen Objektnamen einer Ladeklasse enthalten, die verwendet werden soll, um die angegebene Klasse zu laden.

Beispiel

```
JMX_CREATE_MBEAN CLASSNAME=com.uc4.mbeans.DatabaseExample,
NAME=UC4:type=DatabaseExample
```

Siehe auch:

AE JCL für JMX

5.4 JMX_GET_AGENT

Ermittelt die Agent ID des MBean Servers.Ermittelt die Agent ID des MBean Servers.

Syntax

JMX_GET_AGENT [ALL=...]

Syntaxteil	Beschreibung/Format
ALL=	Auflistung der Agent IDs Mögliche Werte: "YES" und "NO" (Standardwert) "YES" - Listet alle Agent IDs der laufenden MBean Server auf "NO" - Ermittelt die Agent ID des vorhandenen MBean Servers

Beispiel

```
JMX_GET_AGENT ALL=YES
```

Siehe auch:

AE JCL für JMX

5.5 JMX_GET_ATTRIBUTE

Liefert den Wert eines oder aller Attribute eines MBeans

Syntax

JMX_GET_ATTRIBUTE MBEAN=... [, ATTRIBUTE=...]

Syntaxteil	Beschreibung/Format
MBEAN=	Objektname des MBeans.
ATTRIBUTE=	Name des Attributes.

Anmerkungen

Geben Sie ein bestimmtes Attribut an dessen Wert Sie erfahren möchten. Lassen Sie hingegeben den Parameter weg, so liefert die Funktion die Werte aller Attribute des MBeans.

Der Administrator legt mit dem Parameter search_all= in der INI-Datei des Agenten fest, ob die Funktion das MBean auf allen lokalen MBean-Servern sucht oder nur auf jenem mit dem der Agent verbunden ist.

Beispiel

```
JMX_GET_ATTRIBUTE
MBEAN="Catalina:type=GlobalRequestProcessor,name=http-80",
ATTRIBUTE=processingTime
```

Siehe auch:

AE JCL für JMX

5.6 JMX_GET_INFO

Gibt Informationen über ein MBean aus.

Syntax

JMX_GET_INFO MBEAN=... [SHORTINFO=...]

Syntaxteil	Beschreibung/Format
MBEAN=	Objektname des MBeans.
SHORTINFO=	Umfang der Informationen. Erlaubte Werte: "YES", "NO" (Standardwert) "YES" - Nur die Kurzinformationen werden ermittelt. "NO" - Die gesamten Informationen werden ausgegeben.

Anmerkungen

Die Beschreibung des MBeans sowie eine Liste der Attribute, Operationen und Benachrichtigungen wird ermittelt.

Der Administrator legt mit dem Parameter search_all= in der INI-Datei des Agenten fest, ob die Funktion das MBean auf allen lokalen MBean-Servern sucht oder nur auf jenem mit dem der Agent verbunden ist.

Beispiel

```
JMX_GET_INFO MBEAN="Catalina:type=GlobalRequestProcessor,name=http-80"
```

Siehe auch:

AE JCL für JMX

5.7 JMX_INVOKE

Ruft eine Methode eines MBeans auf.

Syntax

JMX_INVOKE MBEAN=..., OPERATIONNAME=..., PARAMS=... [, SIGNATURE=...]

Syntaxteil	Beschreibung/Format
MBEAN=	Objektname des MBeans.
OPERATIONNAME=	Name der Operation, die ausgeführt werden soll.
PARAMS=	Liste der Parameter. Mehrere Parameter sind mit Komma zu trennen.
SIGNATURE=	Falls das angegebene MBean mehrere Methoden mit dem Namen OPERATIONNAME= besitzt, muss mit diesem Parameter die Signatur der auszuführenden Methode übergeben werden.

Anmerkungen

Das Script-Sprachmittel ruft die Funktion eines MBeans auf und übergibt ihr Parameter. Besitzt das MBean mehr als eine Funktion, so können Sie mit SIGNATURE= eine bestimmte Methode ansprechen.

Der Administrator legt mit dem Parameter search_all= in der INI-Datei des Agenten fest, ob die Funktion das MBean auf allen lokalen MBean-Servern sucht oder nur auf jenem mit dem der Agent verbunden ist.

Beispiel

```
JMX_INVOKE MBEAN=UC4:type=DatabaseExample,
OPERATIONNAME=selectInvoice, PARAMS="1996-07-04,1996-07-10",
SIGNATURE="java.lang.String,java.lang.String"
```

Siehe auch:

AE JCL für JMX
Beispielsammlung
Aufruf eines MBeans

5.8 JMX_QUERY_NAMES

Listet registrierte MBeans auf.

Syntax

JMX_QUERY_NAMES [NAME=...] [, NOTFOUND=...] [, MAXROWS=...]

Syntaxteil	Beschreibung/Format
NAME=	Objektname eines bestimmten MBeans oder Filterangabe für mehrere MBeans. Die Wildcardzeichen "*" und "?" können hierbei verwendet werden. "*" steht dabei für beliebige, "?" für genau ein Zeichen.
NOTFOUND=	Behandlung, wenn keine MBeans gefunden wurden. Erlaubte Werte: "NORMAL" oder "ABORT" (Standardwert) "NORMAL" = Der AE-Job läuft weiter. "ABORT" = der AE-Job endet abnormal.
MAXROWS=	Anzahl an Zeilen, die maximal aufgelistet werden sollen. Geben Sie diesen Parameter nicht an, so liefert der Agent alle registrierten MBeans.

Anmerkungen

Für die Filterangabe gibt es JMX-spezifische Regeln zu beachten. Die Wildcards können in der Domain-Bezeichnung verwendet werden. Sie sind auch bei den Eigenschaften zulässig sofern Sie folgendes berücksichtigen. Das Wildcardzeichen "*" ersetzt ein komplettes *Eigenschaft=Wert* Paar (z.B. description=Printer). Wildcards können jedoch nicht innerhalb eines Eigenschaftsnamens oder Wertes angegeben werden (z.B. description=*).

Beispiele für gültige Filter:

"*:*"
"DefaultDomain:*"
"???Domain:description=Printer,*"
"*:description=Printer,type=laser,*"

Beispiel

```
JMX_QUERY_NAMES NAME=JMImplementation:*, NOTFOUND=ABORT
```

Siehe auch:

AE JCL für JMX

5.9 JMX_SET_ATTRIBUTE

Setzt ein Attribut eines MBeans.

Syntax

JMX_SET_ATTRIBUTE MBEAN=..., ATTRIBUTE=..., VALUE=...

Syntaxteil	Beschreibung/Format
MBEAN=	Objektname des MBeans.
ATTRIBUTE=	Name des Attributes.
VALUE=	Wert, der für das Attribut gesetzt werden soll.

Anmerkungen

Der Administrator legt mit dem Parameter search_all= in der INI-Datei des Agenten fest, ob die Funktion das MBean auf allen lokalen MBean-Servern sucht oder nur auf jenem mit dem der Agent verbunden ist.

Beispiel

```
JMX_SET_ATTRIBUTE ATTRIBUTE=State, MBEAN=UC4:type=StandardMBean,
VALUE=running
```

Siehe auch:

AE JCL für JMX

5.10 JMX_UNREGISTER_MBEAN

Entfernt ein bereits registriertes MBean.

Syntax

JMX_UNREGISTER_MBEAN MBEAN=...

Syntaxteil	Beschreibung/Format
MBEAN=	Objektname des MBeans.

Beispiel

```
JMX_UNREGISTER_MBEAN MBEAN=UC4:type=StandardMBean
```

Siehe auch:

AE JCL für JMX

5.11 JMX_WAIT_FOR_NOTIFICATION

Wartet auf eine Benachrichtigung des MBeans.

Syntax

JMX_WAIT_FOR_NOTIFICATION MBEAN=... [, ENABLE=...] [, FILTER=...]

Syntaxteil	Beschreibung/Format
MBEAN=	Objektname des MBeans.
ENABLE=	Liste der Attribute bzw. Benachrichtigungstypen. Mehrere Parameter sind mit Komma zu trennen.
FILTER=	Filter für die Benachrichtigungen. Erlaubte Werte: "NONE" (Standardwert), "ATTRIBUTE" und "TYPE". "NONE" - Alle Benachrichtigungen werden akzeptiert. "ATTRIBUTE" - Es erfolgt eine Filterung auf Attributänderungen. "TYPE" - Es erfolgt eine Filterung nach dem Benachrichtigungstyp. Die Attribute bzw. Typen werden mit dem Parameter ENABLE= angegeben.

Beispiel

```
JMX_WAIT_FOR_NOTIFICATION
MBEAN=JMImplementation:type=MbeanServerDelegate,
ENABLE="JMX.mbean.registered,JMX.mbean.unregistered", FILTER=NONE
```

Siehe auch:

AE JCL für JMX

6 AE JCL für SQL

6.1 Allgemeines zur SQL-JCL

Die Automation Engine stellt für SQL-Jobs spezielle Script-Sprachmittel zur Verfügung. Sie schreiben ihre Ergebnisse in den Report. Von dort sind sie mittels PREP_PROCESS_REPORT auslesbar.

Script-Sprachmittel

Script-Sprachmittel	Beschreibung
SQL_EXECUTE_JOB	Führt einen Job am MS SQL Server durch
SQL_GET_COLUMNS	Liefert Informationen über die Spalten einer Tabelle
SQL_GET_JOBS	Liefert eine Liste aller definierten Jobs eines MS SQL Servers
SQL_GET_TABLES	Listet alle Tabellen der Datenbank auf
SQL_ON_ERROR	Legt die Reaktion auf SQL-Fehler fest
SQL_ON_ROWCOUNT_ZERO	Setzt den Rückgabewert, wenn eine SQL-Anweisung keine Treffer liefert
SQL_SET_STATEMENT_TERMINATOR	Legt das Zeichen für die Trennung der SQL-Anweisungen fest

Siehe auch:

Registerkarte **Form** (SQL)

6.2 SQL_EXECUTE_JOB

Führt einen Job am MS SQL Server durch.

Syntax

SQL_EXECUTE_JOB JOB=...

Syntaxteil	Beschreibung/Format
JOB=	Name des MS SQL Jobs

Anmerkungen

Der Agent verwendet zum Starten der MS SQL Jobs die Stored Procedure sp_start_job in der Datenbank msdb.

Der Status des AE-Jobs hängt von der Durchführung des gestarteten Jobs ab:

Status des MS SQL Jobs	Status des AE-Jobs
MS SQL Job wurde erfolgreich beendet	ENDED_OK
MS SQL Job ist abgebrochen	ENDED_NOT_OK
MS SQL Job wurde vom Anwender abgebrochen	ENDED_CANCEL

Der AE-Job bricht außerdem ab, wenn der Benutzer keine Berechtigung hat oder als Datenbank nicht Microsoft SQL Server verwendet wurde.

Wenn Sie den AE-Job selbst abbrechen, während der MS SQL Job noch aktiv ist, so ruft der Agent die Stored Procedure sp_stop_job auf. Der MS SQL Job bricht daher nicht unbedingt sofort ab und wird solange weiter überwacht bis sein Ende feststeht.

Das Script-Sprachmittel SQL_GET_JOBS listet Ihnen die vorhandenen MS SQL Job auf.

Beispiel

```
SQL_EXECUTE_JOB JOB="Integrity Checks Job for DB Maintenance Plan";
```

Siehe auch:

AE JCL für SQL

6.3 SQL_GET_COLUMNS

Liefert Informationen über die Spalten einer Tabelle.

Syntax

SQL_GET_COLUMNS TABLE=...

Syntaxteil	Beschreibung/Format
TABLE=	Name der Tabelle

Anmerkungen

Der Agent schreibt das Ergebnis in den Report. Pro Spalte wird genau eine Zeile ausgegeben, wobei die einzelnen Werte mit Strichpunkt voneinander getrennt sind.

Aufbau der Information einer Spalte:

Fremdschlüssel;*Primärschlüssel*;*Spaltenname*;*(Datentyp)*

Wert	Beschreibung
Fremdschlüssel	Kennzeichen für den Schlüssel Erlaubte Werte: "0" und "1" "0" - Die Spalte stellt keinen Fremdschlüssel dar. "1" - Die Spalte ist ein Fremdschlüssel.

Primärschlüssel	Kennzeichen für den Schlüssel Erlaubte Werte: "0" und "1" "0" - Die Spalte stellt keinen Primärschlüssel dar. "1" - Die Spalte ist ein Primärschlüssel.
Spaltenname	Name der Spalte
Datentyp	Setzt sich zusammen aus dem herstellerspezifischen Datentyp und der Information ob dieses Feld den Wert NULL erlaubt. Bei Zeichentypen wird nach dem Datentyp die Länge in Klammern angegeben.

⚠ Vor der Ausführung der Script-Funktion erfolgt automatisch ein COMMIT.

Beispiel

Der folgende Aufruf ermittelt die Spalteninformationen zur Tabelle "Products".

```
SQL_GET_COLUMS TABLE="Products";
```

Ergebnis im Report:

```
0;1;ProductID;(int identity, not null)
0;0;ProductName;(nvarchar(40), not null)
1;0;SupplierID;(int, null)
1;0;CategoryID;(int, null)
0;0;QuantityPerUnit;(nvarchar(20), null)
0;0;UnitPrice;(money, null)
0;0;UnitsInStock;(smallint, null)
0;0;UnitsOnOrder;(smallint, null)
0;0;ReorderLevel;(smallint, null)
0;0;Discontinued;(bit, not null)
```

Siehe auch:

AE JCL für SQL

6.4 SQL_GET_JOBS

Liefert eine Liste aller definierten Jobs eines MS SQL Servers.

Syntax

SQL_GET_JOBS

Anmerkungen

Der Agent verwendet für die Ermittlung der MS SQL Jobs die Stored Procedure sp_help_job in der Datenbank msdb. Im Report wird jeder MS SQL Job in einer eigenen Zeile ausgegeben.

Der AE-Job bricht ab, wenn der Benutzer keine Berechtigung hat oder als Datenbank nicht Microsoft SQL Server verwendet wurde.

Das Script-Sprachmittel SQL_EXECUTE_JOB ist in der Lage die MS SQL Jobs auszuführen.

Beispiel

```
SQL_GET_JOBS;
```

Siehe auch:

AE JCL für SQL

6.5 SQL_GET_TABLES

Listet alle Tabellen der Datenbank auf.

Syntax

SQL_GET_TABLES

Anmerkungen

Der Agent schreibt das Ergebnis in den Report. Pro Tabelle wird genau eine Zeile ausgegeben. Falls ein Schema vorhanden ist, wird es vor dem Tabellennamen geschrieben. Schema und Tabelle sind sodann durch einen Punkt getrennt.

Beispiel

Hier werden alle Tabellen der Datenbank "Northwind" ermittelt.

```
SQL_GET_TABLES;
```

Ergebnis im Report:

```
dbo.Categories
dbo.CustomerCustomerDemo
dbo.CustomerDemographics
dbo.Customers
dbo.Employees
dbo.EmployeeTerritories
dbo.Order Details
dbo.Orders
dbo.Products
dbo.Region
dbo.Suppliers
dbo.Territories
```

Siehe auch:

AE JCL für SQL

6.6 SQL_ON_ERROR

Legt die Reaktion auf SQL-Fehler fest.

Syntax

SQL_ON_ERROR ACTION=...

Syntaxteil	Beschreibung/Format
ACTION=	Reaktion auf SQL-Fehler Erlaubte Werte: "ABEND" (Standardwert) und "RESUME" "ABEND" - Beim ersten auftretenden SQL-Fehler bricht der Job mit dem Rückgabewert "1" ab. Nachfolgende SQL-Anweisungen werden nicht ausgeführt. "RESUME" - Bei auftretenden SQL-Fehlern wird der Job fortgesetzt. Der Wert muss unter Hochkomma stehen. Geben Sie den Parameter nicht an, so wird "RESUME" verwendet.

Anmerkungen

Standardmäßig werden Jobs abgebrochen, wenn SQL-Fehler auftreten. Mit diesem Script-Sprachmittel können Sie dieses Verhalten steuern.

Die Einstellung, die Sie mit SQL_ON_ERROR setzen, gilt für alle folgenden SQL-Anweisungen bis zum Job-Ende oder bis zur nächsten SQL_ON_ERROR-Anweisung.

Beachten Sie, dass beim Abbruch des Jobs alle SQL-Anweisungen bis zum letzten COMMIT rückgängig gemacht werden. Enthält der Job kein COMMIT, so werden daher alle SQL-Anweisungen zurückgerollt.

Beispiel

Die zweite INSERT-Anweisung erzeugt einen Fehler, da im Tabellenname ein Tippfehler passiert ist. Trotzdem setzt der Agent den Job fort und kann den Datensatz für Hrn. Huber in die Datenbank schreiben.

```
SQL_ON_ERROR ACTION="RESUME"
insert into person values (1,'Meier');
insert into persson values (2,'Mueller');
insert into person values (3,'Huber');
```

Im folgenden Beispiel bricht der Agent den Job aufgrund des falschen Tabellennamens ab. Durch das COMMIT wird der Eintrag des Datensatzes für Hrn. Meier nicht rückgängig gemacht. Da der Job jedoch bei der zweiten INSERT-Anweisung abgebrochen ist, wird der Datensatz für Hrn. Huber nicht mehr berücksichtigt, obwohl er syntaktisch richtig wäre.

```
SQL_ON_ERROR ACTION="ABEND"
insert into person values (1,'Meier');
COMMIT;
insert into persson values (2,'Mueller');
insert into person values (3,'Huber');
```

Siehe auch:

AE JCL für SQL

6.7 SQL_ON_ROWCOUNT_ZERO

Setzt den Rückgabewert, wenn eine SQL-Anweisung keine Treffer liefert.

Syntax

SQL_ON_ROWCOUNT_ZERO RETCODE=...

Syntaxteil	Beschreibung/Format
RETCODE=	Rückgabewert Standardwert: "1"

Anmerkungen

Mit diesem Script-Sprachmittel können Sie den Job mit einem definierten Rückgabewert abbrechen, wenn eine SQL-Anweisung wie z.B: SELECT oder UPDATE keine Treffer liefert. Die Einstellung, die Sie mit SQL_ON_ROWCOUNT_ZERO setzen, gilt für alle folgenden SQL-Anweisungen bis zum Job-Ende oder bis zur nächsten SQL_ON_ROWCOUNT_ZERO-Anweisung.

Setzen sie den RETCODE auf "0", so wird die Trefferanzahl bei SQL-Anweisungen ignoriert und der Job fortgesetzt.

⚠ SQL_ON_ROWCOUNT_ZERO kann nur für SELECT, UPDATE, INSERT und DELETE eingesetzt werden. Andere SQL-Anweisungen wie z.B: CREATE TABLE liefern auch im Normalfall keine Treffer. Wenn Sie SQL_ON_ROWCOUNT_ZERO im Job verwenden, müssen Sie daher jeweils vor den genannten SQL-Anweisungen daher immer RETCODE auf den Wert "0" setzen.

ℹ Beim Abbruch des Jobs werden alle SQL-Anweisungen bis zum letzten COMMIT rückgängig gemacht.

Beispiel

Liefert eine der drei SQL-Anweisungen "0" Treffer zurück, so wird der Job mit dem Rückgabewert "1" abgebrochen.

```
SQL_ON_ROWCOUNT_ZERO RETCODE=1;
select * from person;
insert into person values (1,'Meier');
update person set adresse = 'Hauptstraße 8'
where name = 'Huber';
```

Siehe auch:

AE JCL für SQL

6.8 SQL_SET_STATEMENT_TERMINATOR

Legt das Zeichen für die Trennung der SQL-Anweisungen fest.

Syntax

SQL_SET_STATEMENT_TERMINATOR TERM=...

Syntaxteil	Beschreibung/Format
TERM=	Trennungszeichen Standardwert: ";"

Anmerkungen

⚠ Beachten Sie, dass diese Funktion automatisch in das Script eingetragen wird, wenn das Trennungszeichen innerhalb einer SQL-Anweisung vorkommt. Es ist somit nicht notwendig, dass Sie die Funktion selbst einfügen!

Bei der Auswahl eines Trennungszeichens wird die folgende Reihenfolge eingehalten:

";", "@", "$", "/", "~", "*", "+", "?", "=", ".", "-", "§", "ë"

Die Einstellung, die mit SQL_SET_STATEMENT_TERMINATOR gesetzt wird, gilt für alle folgenden SQL-Anweisungen bis zum Job-Ende oder bis zur nächsten SQL_SET_STATEMENT_TERMINATOR-Anweisung.

Beispiel

Da innerhalb einer SQL-Anweisung ";" vorkommt, wird SQL_SET_STATEMENT_TERMINATOR im Script eingefügt.

```
SQL_SET_STATEMENT_TERMINATOR TERM='@';

DECLARE
v_einheit_kurz varchar2(10);
v_bezeichnung varchar2(40);
BEGIN
v_einheit_kurz := 'kg';
v_bezeichnung := 'Kilogramm';
insert into einheit (einheit_kurz, bezeichnung)
values (v_einheit_kurz, v_bezeichnung);
EXCEPTION when DUP_VAL_ON_INDEX then
/* Der Datensatz ist schon da => Bezeichnung neu setzen */
update einheit set
bezeichnung = v_bezeichnung
where einheit_kurz = v_einheit_kurz;
END;

@
```

Siehe auch:

AE JCL für SQL

Glossar

Dieses Glossar enthält spezifische Fachbegriffe.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

.1 A

- **AE-Datenbank**
  Ein relationales Datenbank-Management-System (RDMS), das alle Scheduling-Daten zentral verwaltet. Es enthält Objektdefinitionen, Systemeinstellungen, Statistikdaten, Jobreports, etc.
- **AE-Komponenten**
  Dazu gehören alle Programme der Automation Engine wie z.B. UserInterfaces, AutomationEngine, Agenten, ServiceManager, Dienstprogramme usw.
- **AE-Priorität**
  Nimmt Einfluss auf die Verarbeitung der Aufgaben innerhalb eines AE-Systems.
- **AE-Script**
  Script-Sprache vder Automation Engine.
- **AE-System**
  Umgebung, die durch AE-Komponenten gesteuert wird.
- **AE-Variablen**
  Variablen-Objekte, die Einstellungen für das AE-System enthalten.
- **AblaufPlan**
  Veralteter Begriff. Ersetzt durch: Workflow
- **Agent**
  Programm, das die Ausführung von Verarbeitungen auf Zielsystemen wie z.B. Rechner oder Geschäftslösungen ermöglicht. Auch ein eigener Objekttyp in der AE.
- **AgentenGruppe**
  Eine AgentenGruppe fasst Agenten einer Plattform zusammen. Welche Agenten in einer AgentenGruppe aufgenommen werden sollen, bestimmen Sie indem Sie entweder die konkreten Namen eintragen oder Filter verwenden. Eine Aufgabe, die im Rahmen einer AgentenGruppe läuft, wird abhängig vom Modus auf einem oder allen Agenten der AgentenGruppe durchgeführt.
- **Aktivierung**
  Durch die Aktivierung erhält die Aufgabe eine RunID, erscheint im Aktivitätenfenster und ist für die Durchführung bereit. (siehe auch Start)
- **Aktivierungsprotokoll**
  Bericht, der Details zur Aktivierung der Aufgabe enthält. Welche Details in das Protokoll aufgenommen werden, hängt von den gesetzten Einstellungen ab (z.B. generierte JCL, Änderungen von Variablen, usw.).
- **Aktivität**
  siehe Aufgabe
- **Aktivitätenfenster**
  Fenster im UserInterface, das die aktivierten Objekte anzeigt.
- **Alias**
  Name für Aufgaben, der statt dem tatsächlichen Objektnamen im Aktivitätenfenster, Monitor

und Statistik angezeigt wird. Möglich für Workflow-Aufgaben und Objekte die einmalig, periodisch oder über das Script aktiviert werden.

- **Arbeitsprozess**
  Teil der Komponente Automation Engine. Kümmert sich um die Verarbeitung in einem UC4-System (siehe auch primärer Arbeitsprozess).
- **Aufgabe**
  Ein gestartetes Objekt, welches gerade durchgeführt wird. Aufgaben werden auch als Aktivitäten bzw. Tasks bezeichnet.
- **Automation Engine**
  Diese Komponente steuert ein AE-System. Besteht aus verschiedenen Serverprozessen.
- **Automation Engine**
  Eigenes Automic Produkt, welches zur Steuerung, Verwaltung und Betrieb eines AE-Systems dient. Ermöglicht die Definition und Einplanung von Objekten, welche Prozesse und Tätigkeiten auf verschiedenen Host durchführen.
- **Autoprognose**
  Zeigt jene Aufgaben an, die in einem vorgegebenen Zeitraum laufen werden, und bietet somit eine umfassende Vorschau auf zukünftige Aktivitäten.

.2 B

- **Batchmodus**
  Sequentielle Hintergrundverarbeitung von Prozessen.
- **Benachrichtigung**
  Sendet Mitteilungen an einzelne Benutzer und BenutzerGruppen des Automation Engine (AE)-Systems. Auch ein eigener Objekttyp in der AE.
- **Benachrichtigungs-Monitor**
  Fenster der Benachrichtigung, die zur Laufzeit an ein oder mehrere Benutzer gesendet wird.
- **Benutzer**
  Person, die ein UC4-System nutzt. Auch ein eigener Objekttyp in UC4.
- **BenutzerGruppe**
  Zusammenfassung von Benutzern denen ein gemeinsames Rechteprofil zugewiesen werden soll. Auch ein eigener Objekttyp in der Automation Engine. BenutzerGruppen dienen der einfacheren Handhabung von mehreren Benutzern. Anstatt jedem Benutzer einzeln Zugriffsrechte zuweisen zu müssen, können so einer Gruppe auf einmal die gleichen Zugriffsrechte zugewiesen werden. Die Handhabung von Benutzern wird dadurch effizienter, aber auch sicherer, weil durch die BenutzerGruppen ein besserer Überblick über die zugewiesenen Rechte gewährleistet wird.

.3 C

- **CallAPI**
  Programmierschnittstelle, die entweder direkt oder aus einem anderen Programm aufgerufen werden kann, und ein Script im AE-System ausführt.
- **CallOperator**
  Veralteter Begriff. Ersetzt durch: Benachrichtigung

- **Child / Children**
  Objekte, die von übergeordneten Aufgaben (Parent) aktiviert werden.
- **Client Queue**
  Queue-Objekt, welches in jedem Mandanten standardmäßig vorhanden ist.
- **Cockpit**
  Visualisiert Werte und Zustände von der AE oder von den überwachten und gesteuerten Systemen. Auch ein eigener Objekttyp in der AE.
- **CodeTabelle**
  Definiert einen kompletten Zeichensatz. Auch ein eigener Objekttyp in der AE.

.4 D

- **Datensequenz**
  Interne Auflistung (zum Beispiel: Konsolenausgaben oder Zeilen eines Variablen-Objektes), auf deren Zeilen mit einer PROCESS-Schleife und dem Sprachmittel GET_PROCESS_ LINE zugegriffen werden kann. Datensequenzen werden mit den PREP_PROCESS*-Sprachmitteln erzeugt.
- **DialogClient**
  Veralteter Begriff. Ersetzt durch: UserInterface
- **Dialogprozess**
  Teil der Komponente Automation Engine und Sonderform des Arbeitsprozesses. Kümmert sich nur um Nachrichten, die von UserInterfaces stammen.
- **Dienstprogramme**
  Unterstützen die Verwaltungsaufgaben eines UC4-Systems wie z.B. die Reorganisation und Archivierung der UC4 Datenbank.
- **Dynamische Variable**
  Variablen-Objekt mit dem Attribut "Quelle" - "SQL", "SQL -intern", "Multi" oder "Dateiliste". Werte werden bei jedem Zugriff direkt von der Datenquelle geholt und nicht im Objekt gespeichert.

.5 E

- **E-Mail-Kopplung**
  Funktionalität von Windows- und UNIX-Agenten, die das Senden von E-Mails ermöglicht.
- **Enterprise Control Center**
  Eigenes Automic-Produkt. Webanwendung, welche den übersichtlichen und einfach Zugriff auf Funktionen verschiedener Automiation Engine-Anwendungen / -Produkte ermöglicht. Verfügbar über das Automic Download Center.
- **Ereignis**
  Aktion, die ausgelöst wird, wenn bestimmte Bedingungen zutreffen. Auch ein eigener Objekttyp in der Automation engine.
- **Event-ID**
  Erste RunID von Dateisystem- und Konsol-Ereignissen. Beide Ereignistypen erfordern eine Kommunikation zwischen der Komponente Automation Engine und Agent. Damit auch nach

---

einem Logwechsel und der damit verbundenen Änderung der RunID eine Zuordnung zum Ereignis noch möglich ist, erfolgt die Kommunikation über die erste RunID.

- **Executor**
  Veralteter Begriff. Ersetzt durch: Agent
- **Explorer**
  Fenster im UserInterface in dem Objekte angelegt, bearbeitet und verwaltet werden.
- **externe Abhängigkeit**
  Aufgabe, deren Endstatus im Ablauf eines Workflows berücksichtigt wird, die jedoch selbst nicht im Rahmen des Workflows läuft.

.6 F

- **FileTransfer**
  Überträgt Dateien zwischen zwei Rechnern. Auch ein eigener Objekttyp in der Automation Engine.

.7 G

- **Graphical Workload Analyzer**
  Veralteter Begriff. Ersetzt durch: UC4 ClearView
- **Gruppe**
  Fasst Aufgaben zusammen, um diese gemeinsam durchzuführen. Auch ein eigener Objekttyp in der Automation Engine.
- **Gruppen-Monitor**
  Fenster, welches den Zustand der Aufgaben einer Gruppe anzeigt.

.8 H

- **HTML-Help**
  Hilfeformat von Microsoft für Handbücher. Die Hilfedateien enden mit .CHM. (siehe auch WebHelp)
- **Host**
  Rechner, Zielsystem.
- **Hostattribute**
  Attribute des Job-Objektes, die plattformabhängig sind.

.9 I

- **Include**
  Script, das häufig und in mehreren Objekten verwendet wird. Auch ein eigener Objekttyp in der Automation Engine.

.10 J

- **Job**
  Verarbeitung auf einem Zielsystem. Auch ein eigener Objekttyp in der Automation Engine.
- **Job Control Language**
  Abkürzung für "Job Control Language". Damit sind Anweisungen gemeint, die Verarbeitungsschritte bilden und auf Rechnern ausgeführt werden.
- **JobPlan**
  Veralteter Begriff. Ersetzt durch: Workflow

.11 K

- **Kalender**
  Fasst Tage in Kalenderbegriffen zusammen. Auch ein eigener Objekttyp in der AE.
- **Kalenderbedingung**
  Durchführungskriterium für Aufgaben, die auf Kalenderbegriffen basieren.
- **Kalenderbegriff**
  Teil eines Kalender-Objektes, in dem Tage definiert werden.
- **Key-Spalte**
  Spalte bei statischen Variablen-Objekten, über die auf Werte einer bestimmte Zeile zugegriffen werden kann.
- **Kommunikationsprozess**
  Teil der Komponente Automation Engine. Kommunikationsprozesse bilden die Verbindung zu AE-Komponenten.

.12 L

- **Laufzeit**
  Durchführungsdauer einer Aufgabe. Das ist der Zeitraum zwischen Start und Ende. Der Aktivierungszeitraum wird nicht dazugerechnet (siehe auch Aktivierung und Start).
- **Login**
  Anmeldedaten für Zielsysteme. Auch ein eigener Objekttyp in der AE.

- **logisches Datum**
  Das logische Datum wird als Vergleichsdatum bei der Prüfung von Kalenderbedingungen verwendet.

.13 M

- **Mandant**
  Abgeschlossene Umgebung für die Durchführung von Aufgaben innerhalb eines AE-Systems. Auch ein eigener Objekttyp in der AE.
- **Meldungsfenster**
  Fenster im UserInterface, das Warnungen, Informations- und Fehlermeldungen ausgibt.

.14 N

- **Nonstop-Prozesse**
  Teil der Komponente Automation Engine. Nonstop-Prozesse übernehmen die Verarbeitung, wenn der Rechner mit den aktiven Serverprozessen ausfällt.

.15 O

- **Objekt**
  Tätigkeiten und Abläufe, welche die Automation engine steuert, werden mit Hilfe von sogenannten Objekten abgebildet (siehe auch Aufgabe).
- **Objektklasse**
  Alle Objekte werden in 4 Klassen unterteilt: aktivierbare Objekte, aktive Objekte, passive Objekte und Systemobjekte.
- **Objekttyp**
  Für jede Tätigkeit existiert ein eigenes Objekt: Agent, Benachrichtigung, Benutzer, BenutzerGruppe, Cockpit, CodeTabelle, Dokumentation, Ereignis, FileTransfer, Gruppe, Include, Job, Workflow, Kalender, Login, Mandant, RemoteTaskManager, Schedule, Script, Server, Sync, Variable und ZeitZone.
- **Objektvariablen**
  Platzhalter für Werte, die einem Objekt in dessen Registerkarte "Variablen & Prompts" hinterlegt werden.

.16 P

- **Parent**
  Ein Objekt kann auf verschiedenste Arten aktiviert werden. Der Verursacher der Aktivierung wird als übergeordnete Aufgabe (Parent) bezeichnet. (siehe auch Child, Children)
- **Periodencontainer**
  Steuert die Durchführung von periodischen Aufgaben.
- **Periodische Aufgaben**
  Werden ohne Schedule-Objekt eingeplant und weisen zumeist eine Periode, die kleiner als ein Tag ist, auf.
- **Perspektive**
  Eigener Funktionsbereich der Enterprise Control Center (ECC) - Weboberfläche. Die Perspektiven "Service Catalog", "Process Monitoring" und "Process Assembly" stellen Funktionalitäten der Automation Platform bereit.
- **Predictive Analytics**
  Eigenes UC4-Produkt. Komplexes, graphisches Analysetool, das verschiedene, als Visualisierungen bezeichnete, interaktive grafische Darstellungen eines Datensatzes erzeugt. Abhängig von den verwendeten Funktionen, können die Visualisierungen sowohl einzelne Datenwerte als auch Aggregationen zeigen. Wenn die spezielle eventBase für SLA-Ergebnisdaten in Ihrem Unternehmen verwendet wird, können Sie erweiterte Analysen abrufen und anzeigen, die Muster und Trends in den Verlauf-Daten zur SLA-Leistung aufzeigen. Verfügbar über das UC4 Download Center.
- **Process Assembly**
  Perspektive des Enterprise Control Centers. Ermöglicht die Erstellung, Definition und Änderung von Workflows.
- **Process Automation**
  Alter Name der Perspektive Service Catalog.
- **Process Monitoring**
  Perspektive des Enterprise Control Centers. Listet die Aktivitäten aller Benutzer auf und ermöglicht diese zu beeinflussen (abbrechen, deaktivieren, ...).
- **Prognose**
  Einschätzung der Laufzeit einer Aufgabe, die auf den vorangegangenen Durchführungen basiert.
- **PromptSet**
  Benutzerdefinierte Eingabemaske für aktivierbare Objekte. Auch ein eigener Objekttyp in der AE.
- **PromptSet-Element**
  Felder / Steuerelemente, mit deren Hilfe Werte vom Benutzer abgefragt werden. Bilden den Inhalt einer PromptSet-Eingabemaske.
- **PromptSet-Variable**
  Speichert den Wert eines PromptSet-Elementes. Dabei kann es sich je nach Situation um die vom Benutzer eingegeben oder um die Standardwerte handeln. Verhalten sich wie Objektvariablen.
- **primärer Arbeitsprozess**
  Führt zentrale Aufgaben der Arbeitsprozesse durch, die keine Verteilung erlauben (Zeitbasis, Verwaltung der Prozesse etc.).

.17 Q

- **Queue**
  Objekttyp in der AE. Legt die maximale Anzahl parallel laufender Aufgaben, deren Prioritäten und somit die Reihenfolge von auszuführenden Objekten fest.
- **QueueManager**
  Veralteter Begriff. Ersetzt durch: RemoteTaskManager

.18 R

- **RA Solution**
  Lösung basierend auf der Rapid Automation Technologie, die der AE den Zugriff auf ein externes System / Anwendung / Platfform ermöglicht. Dabei handelt es sich um eine JAR-Datei, welche in die AE-Datenbank zu laden und mit einem RA-Agenten zu verknüpfen ist. Nach dem Laden der Solution stehen die spezifischen RA-Objekte (Jobs, Verbindungen, Agent) im AE-System zur Verfügung.
- **RA-Agent**
  Agent, der mit einer bestimmten RA Solution verknüpft werden kann und so deren Funktionalitäten für ein AE-System zur Verfügung stellt. Er stellt somit die Schnittstelle zwischen einem externen System / Anwendung / Plattform und einem AE-System dar.
- **Rapid Automation**
  Generische Technologie, die verschiedene Lösungen aufnehmen kann. Besteht aus einem RA-Agenten und einer RA-Solution.
- **Release Manager**
  Veralteter Begriff.
- **Release Notes**
  Release Notes informieren Sie über Highlights, neue Funktionen, Verbesserungen und über Korrekturen unserer Versionen und Freigaben aus der Automic Produktfamilie.
- **Release Orchestrator**
  Die Application Release Automation teilt sich auf in die Produkte Release Orchestrator und Deployment Manager.
- **RemoteTaskManager**
  Überwacht und steuert externe Jobs, die nicht von der Automation Engine gestartet worden sind. Auch ein eigener Objekttyp in der AE
- **Report**
  Bericht, der nähere Informationen über die Durchführung einer Aufgabe oder einer AE-Komponente enthält.
- **Result-Spalte**
  Erste Spalte von dynamischen Variablen-Objekten mit der Quelle "SQL", "SQL-intern" und "Multi". Der Inhalt dieser Spalte wird mit dem Result-Format festgelegt.
- **RunID**
  Kurzform für "laufende Nummer". Hierbei handelt es sich genauer um eine Zahl, welche die Durchführung einer Aufgabe eindeutig kennzeichnet. Die RunID kann zwischen 7 und 10 Stellen aufweisen. Sie wird von der Komponente Automation Engine vergeben.
- **Rückgabewert**
  Wert, der das Ergebnis von Aufgaben und Script-Funktionen repräsentiert.
- **reales Datum**
  Datum, welches zur Prüfung der Laufzeitüberwachung oder den Zeitbedingungen in den

Eigenschaften von Workflow-Aufgaben verwendet wird. Das reale Datum entspricht dem Aktivierungszeitpunkt des obersten Workflows und wird an alle untergeordneten Aufgaben vererbt.

.19 S

- **Schedule**
  Startet in periodischen Abständen aktivierbare Objekte. Auch ein eigener Objekttyp in der AE.
- **Schedule-Monitor**
  Graphische Ansicht der Durchführung von Schedule-Objekten.
- **Script**
  Verarbeitungsanweisungen in der Script-Sprache der AE. Auch ein eigener Objekttyp in der AE.
- **Script-Variable**
  Platzhalter eines Wertes in einem Script.
- **Serverprozess**
  Serverprozesse bilden die Grundlage für die Komponente Automation Engine. Unterteilt sich in verschiedene Arten: Kommunikationsprozesse, Arbeitsprozesse, Dialogprozesse und NonStop-Prozesse.
- **Service Catalog**
  Perspektive des Enterprise Control Centers. Ermöglicht dem Benutzer, die Objekte in seinem Favoriten-Ordner zu starten und deren Ausführung grob zu überwachen.
- **Service Orchestrator**
  Automic Produkt. Service Orchestrator ist eine Perspektive (Plug-In) des Enterprise Control Centers (ECC) und dient zum Verwalten, Überwachen und Analysieren der Performance von SLAs (Service Level Agreements).
- **ServiceManager**
  Programm, welches das Starten und Beenden von AE-Komponenten ermöglicht.
- **Statistik**
  Aufstellung über die vergangenen Durchführungen einer Aufgabe.
- **Status**
  Zustand einer Aufgabe (z.B. aktiv, blockiert, in Generierung, usw.).
- **Sub-Workflow**
  Workflow, der Teil eines Workflows ist.
- **Sync**
  Synchronisiert aktivierbare Objekte auf der Grundlage definierter Zustände und Aktionen. Auch ein eigener Objekttyp in der AE.
- **Sync-Monitor**
  Fenster, welches den Zustand eines Sync-Objektes und dessen zugewiesenen Aufgaben anzeigt.
- **Systemübersicht**
  Fenster im UserInterface, das Informationen über das AE-System beinhaltet.
- **statische Variable**
  Variablen-Objekt mit der Einstellung "Quelle" - "Statisch": Variablenwerte werden durch den Benutzer oder per Script eingetragen und bleiben im Objekt gespeichert.

.20 T

- **teilqualifizierter FileTransfer**
  FileTransfers, die mit Hilfe von Wilcards mehrere Dateien übertragen (im Gegensatz zu vollqualifzierten FileTransfers).

.21 U

- **UC4 Automation Engine**
  Eigenes UC4 Produkt. Enthält die einzelnen Komponenten, die benötigt werden, um ein UC4-System zu betreiben (wie z.B. die gleichnamige Komponente Automation Engine, welche die Serverprozesse enthält, das UserInterface, die Agenten, das WebInterface etc.). Verfügbar über das UC4 Download Center.
- **UC4 Automation Platform**
  Veralteter Begriff. Ersetzt durch: UC4 Automation Engine
- **UC4 ClearView**
  Eigenes UC4 Produkt. Graphisches Analysewerkzeug: Stellt Statistik-, Aktivitäten- und Prognose-Daten pro Mandant eines UC4-Systems in einem Balken-Diagramm dar und ermöglicht die Berechung des kritischen Pfades.Verfügbar über das UC4 Download Center.
- **UC4 Decision**
  Veralteter Begriff. Ersetzt durch: UC4 Policy Orchestrator
- **UC4 Insight**
  Veralteter Begriff. Ersetzt durch: Predictive Analytics
- **UC4 ONE Automation 2013**
  Name der UC4-Produktpalette.
- **UC4 Operations Manager**
  Veralteter Begriff. Ersetzt durch: UC4 Automation Platform
- **UC4 Policy Orchestrator**
  Eigenes UC4-Produkt zur Verwaltung von Ereignissen in einem UC4-System. Diese Anwendung ist das Backend für den Policy Orchestrator. Als Erstes stellt UC4 Policy Orchestrator die Bausteine für die Definition von Geschäftsregeln im Policy Orchestrator zur Verfügung. Anschließend überwacht es die eventBase-Objekte von UC4 auf Vorkommen der Situationen, die in den Bedingungen und Ausnahmen der Geschäftsregeln beschrieben werden, und löst die in den Regeln verlangten Aktionen aus. Verfügbar über das UC4 Download Center.
- **UC4 Server**
  Alter Begriff für die Komponente Automation Engine (v8 und niedriger).
- **UC4.DB Archive**
  Das Dienstprogramm UC4.DB Archive dient dazu, den stetig wachsenden Bestand an Daten aus der UC4-Datenbank auszulagern.
- **UC4.DB Change**
  Dienstprogramm zum Ändern von Transportkoffer-Exporten.
- **UC4.DB Client Copy**
  Dienstprogramm zum Löschen und Kopieren von Mandanten.
- **UC4.DB Load**
  Das Dienstprogramm UC4.DB Load lädt Daten zu unterschiedlichsten Zwecken in die UC4 Datenbank.

- **UC4.DB Reorg**
  Dienstprogramm für die Reorganisierung der UC4-Datenbank. Datensätze werden auf Grund bestimmter Einstellungen mit einem Löschkennzeichen versehen.
- **UC4.DB Reporting Tool**
  Das Dienstprogramm UC4.DB Reporting Tool bietet die Möglichkeit, Auswertungen in Ihrem UC4-System vorzunehmen. Das Ergebnis wird als Reportdatei präsentiert.
- **UC4.DB Revision Report**
  Dienstprogramm zur Entladung von Revisionsreports aus der UC4-Datenbank. Revisionsreports enthalten detaillierte Informationen über Objektänderungen und Zugriffe.
- **UC4.DB Unload**
  Dienstprogramm zum Entladen der UC4-Datenbank
- **UC4.Log Mix**
  Das Dienstprogramm UC4.LogMix unterstützt Sie indem es aus mehreren Report oder Log- bzw. Tracedateien eine gemeinsame Datei erzeugt.
- **Universal Time Coordinated**
  Die Automation Engine verwendet intern aufgrund der weltweiten Eindeutigkeit UTC (Universal Time Coordinated). Für die Anzeige der Uhrzeit und deren Verwendung in Aufgaben und Script-Sprachmitteln können entsprechende Zeitzonen-Objekte, die auf die lokale Zeit umrechnen, angelegt werden.
- **UserInterface**
  Graphische Benutzeroberfläche von UC4.

.22 V

- **Variable**
  Speichert oder ermittelt Werte dynamisch zur Laufzeit. Auch ein eigener Objekttyp in UC4.
- **Versionsmanagement**
  Gesicherte Version eines Objektes, das bei Änderungen angelegt wird und später wiederhergestellt werden kann.
- **vollqualifizierter FileTransfer**
  FileTransfers ohne Wildcards. Es wird nur eine bestimmte Datei übertragen (im Gegensatz zu teilqualifizierten FileTransfers)..
- **vordefinierte Variable**
  Fixe Variablen, die in Attributen oder im Script aktivierbarer Objekte verwendet werden können. Die Werte beziehen sich auf das Objekt oder das System.
- **vorgemerkt**
  Status einer Aufgabe, die innerhalb einer Gruppe läuft und auf ihren Start wartet.

.23 W

- **WebHelp**
  Hilfeformat für Handbücher, das mit einem Webbrowser geöffnet werden kann. (siehe auch HTML-Help).
- **WebInterface**
  Benutzeroberfläche von UC4, die über einen Webbrowser aufrufbar ist.
- **Wiederanlauf**
  Wiederholung einer bereits gelaufenen Objekt-Durchführung, wobei einige Besonderheiten gegenüber einem erneuten Start gelten.
- **Wildcardzeichen**
  Platzhalter für Zeichen in Filterangaben (? für genau ein Zeichen, * für beliebig viele Zeichen).
- **Workflow**
  Ablauf von Verarbeitungen. Auch ein eigener Objekttyp in UC4.
- **Workflow-Monitor**
  Graphische Ansicht der Durchführung von aktiven Workflow-Objekten. Zeigt die Workflow-Struktur und den Status der untergeordneten Aufgaben.

.24 X

- **XML-Datei**
  Format des Im- und Exports. Enthält die Struktur von Objekten.

.25 Z

- **Zeitzone**
  Definiert eine lokale Zeit. Auch ein eigener Objekttyp in der AE.